MACHINE LEARNING AND DATA MINING

MACHINE LEARNING AND DATA MINING

REINFORCEMENT LEARNING AND MULTI-AGENT SYSTEMS

Reinforcement Learning in Distributed Domains: Beyond Team Games

David H. Wolpert
NASA Ames Research Center
Moffett Field, CA 94035
dhw@ptolemy.arc.nasa.gov

Joseph Sill
Ripfire, Inc.
San Fransisco, CA 94102
joe@ripfire.com

Kagan Tumer
NASA Ames Research Center
Moffett Field, CA 94035
kagan@ptolemy.arc.nasa.gov

Abstract

Using a distributed algorithm rather than a centralized one can be extremely beneficial in large search problems. In addition, the incorporation of machine learning techniques like Reinforcement Learning (RL) into search algorithms has often been found to improve their performance. In this article we investigate a search algorithm that combines these properties by employing RL in a distributed manner, essentially using the team game approach. We then present bi-utility search, which interleaves our distributed algorithm with (centralized) simulated annealing, by using the distributed algorithm to guide the exploration step of the simulated annealing. We investigate using these algorithms in the domain of minimizing the loss of importance-weighted communication data traversing a constellations of communication satellites. To do this we introduce the idea of running these algorithms "on top" of an underlying, learning-free routing algorithm. They do this by having the actions of the distributed learners be the introduction of virtual "ghost" traffic into the decision-making of the underlying routing algorithm, traffic that "misleads" the routing algorithm in a way that actually improves performance. We find that using our original distributed RL algorithm to set ghost traffic improves performance, and that bi-utility search — a semi-distributed search algorithm that is widely applicable — substantially outperforms both that distributed RL algorithm and (centralized) simulated annealing in our problem domain.

1 Introduction

Use of a centralized algorithm to perform search in large spaces can be quite problematic. This is especially so when the search problem is to find a control policy for a large, distributed Multi-Agent System (MAS) in which communication limitations restrict the amount of data that a centralized algorithm could exploit. Satisfactory performance in such domains may not be possible at all unless one uses a distributed rather than centralized algorithm.

A canonical example of such a control problem, central to mid-term NASA interplanetary missions, is controlling the communication of scientific data up from a planet, across an orbiting constellation of communication satellites orbiting that planet, and thence down to Earth. In addition to its distributed nature, this domain presents a number of other difficulties. One is that the satellites must be able to exercise a large degree of autonomy, since communication with earth can take hours. Another is the need for robustness and adaptability, since the communication loads can be quite unpredictable. Both of these difficulties argue for the use of learning algorithms to perform the distributed control. In particular, the technique of Reinforcement Learning (RL) has often been successfully applied to search algorithms [6; 14; 26], and suggests itself for this domain. More generally, integrating such RL into a distributed algorithm may be highly beneficial for many distributed control problems.

In this article we concentrate on algorithms that achieve this integration by having the agents in the MAS each run RL algorithms. The design problem is how to ensure that as those agents each try to optimize the values of their personal utility functions, an overall "world utility function", quantifying the desirability of the various possible behaviors of the overall system, is maximized. Though most naturally viewed as a means of distributed control, such an approach can actually be used for distributed search in general, by identifying each of the distributed components of the search problem with an agent.

The COllective INtelligence (COIN) framework addresses such design problems [20; 23; 24]. That work has focussed on modifying the agents' personal utility functions so that they induce the best possible value of the world utility function. The learning-based search algorithms we have explored to date based on the COIN framework have been extremely successful. In fact, even in domains where centralized algorithms can be used, due to their adaptability and their exploiting the learn-

ing power of each of the individual agents COIN-based systems typically perform better than such centralized algorithms.

Nonetheless, many centralized algorithms have a number of well-understood strengths, that if integrated with a distributed RL approach like that of COINs might result in performance better than either approach alone. In addition, often search algorithms that are conventionally viewed as centralized can be implemented in a quasi-distributed manner. This opens up the possibility that a hybrid of such an algorithm with a distributed RL algorithm might not only achieve better performance than either alone, but also be implementable in a large distributed problem domain.

In this paper we investigate hybrids of distributed RL algorithms and search algorithms that are (conventionally viewed as) centralized. We concentrate on the use of Simulated Annealing (SA) as the centralized algorithm. SA is a natural choice because its behavior and strengths are very well-understood [1; 4; 5; 13; 16; 17; 21] In addition, at the expense of periodic centralized broadcasts to all agents of whether to choose a new action or repeat their most recent one, and of associated measurements of global performance, SA can be implemented in a quasi-distributed manner. No inter-agent communication is required beyond those broadcasts and measurements.

As our problem domain in this paper we concentrate on the problem of minimizing the importance-weighted loss of scientific data flowing across a constellation of communication satellites. The first contribution of this paper is a novel "baseline" distributed control algorithm for this domain, one that involves no learning. To minimize the number of confounding distinctions between that baseline algorithm and the COIN algorithm we investigate, we have that COIN algorithm "run on top" of the baseline algorithm. More precisely, our second contribution is to use a baseline algorithm in concert with a COIN algorithm in which each agent's action is the determination of fictitious "ghost traffic" that is presented to the baseline algorithm, thereby (hopefully) inducing that baseline algorithm to achieve an even better value of the world utility. (Note that this idea can be applied with most any baseline algorithm and most any distributed RL algorithm.) In this paper we investigate the simplest kind of COIN algorithm, which essentially works by using the team game (TG) approach [8]. Our final contribution is the hybrid of that distributed RL algorithm with SA, a hybrid we call "bi-utility search". This hybrid works by interleaving the Boltzmann-distribution-based exploitation step of conventional SA with a novel exploration step, where rather than have the overall system randomly step away from the current point each agent takes a step that its RL algorithm recommends.

In the next section we discuss the use of distributed RL in the context of our particular problem domain. In Section 3 we define that problem domain in a fully formal manner. Then in Section 4 we present experimental comparisons of TG-based bi-utility search with both TG and SA in that domain. We find that using our original distributed RL algorithm to set ghost traffic improves performance, and that bi-utility search — a semi-distributed search algorithm that is widely applicable — substantially outperforms both that distributed RL algorithm and (centralized) simulated annealing in our problem domain.

2 Background

Many future NASA projects will involve data traversing a constellation of communication satellites. In such a constellation, each satellite continually receives data (e.g., data uplinked from a planet, relayed from another satellite, or even directly observed), and needs to relay this data along a path back to Earth. In general, different data are likely to have different levels of importance. Accordingly, a suitable overall goal for such a constellation to minimize is the total importance-weighted loss of data across the network. We can cast this as maximizing a world utility function given by the negative of that loss. Due to various hardware limitations (e.g., storage, power, bandwidth), even at those times that a particular satellite has a direct link to Earth, to maximize this world utility, it will often still be preferable for that satellite to route its current data across other satellites rather than send it directly to Earth For example, since we are implicitly assuming multiple communication transmitters at each satellite, it may benefit a satellite to drain its current disk by offloading data to other satellites while it is also downloading data to Earth. In this way, the disk will be emptier and therefore better able to accommodate future data surges.

Clearly given the time delays and hardware limitations inherent in this problem, one would like to use as decentralized an algorithm as possible for determining the routing. Unfortunately, while they can be implemented in decentralized forms, traditional routing algorithms (e.g., shortest path algorithms [2; 3; 10]) are ill-suited for our world utility function. Accordingly, in this study we had to first develop a non-learning "baseline" distributed routing algorithm, one that we expect will do well for our world utility. To create a distributed RL version of that algorithm we then consider the use of "ghost" traffic, which is non-existent traffic that is presented along with the real traffic to the baseline routing algorithms running on the satellites. The goal of the learning is to find such ghost traffic that will "fool" the baseline algorithm into having the satellites take better routing decision (as far as world utility is concerned) than they would otherwise.

A natural choice for the type of learning to use to optimize ghost traffic is RL [6; 8; 12; 15; 19; 26]. Indeed, distributed MAS control algorithms in which the agents independently attempt to maximize their personal utilities using RL have been successfully used to optimize world utility in several large decentralized problems [11; 14; 18; 20; 22; 23; 24]. In the TG approaches in par-

ticular [8], every agent uses RL with its utility set to the world utility, so the only centralized communication needed is periodic broadcasts of the world reward signal (in our case presumably calculated on Earth where all the data is collected), a signal shared by all agents. This contrasts with conventional centralized RL, in which the information periodically broadcast has to be "personalized" to each satellite, being that satellite's updated routing policy.

There are a number of major shortcomings of conventional TG however. One of them is that for large systems each agent faces an overwhelming signal-to-noise problem in determining how its actions affect the rewards it receives [25]. Another arises from the fact that each agent's predictions of expected rewards is independent of the other agents' actions, i.e., each agent i considers a distribution of the form:

$$P(utility \mid action_i; observation_i)$$

rather than one of the form

$$P(utility, \mid action_1, action_2, ...; observation_i).$$

Unfortunately, it is quite common that utility is maximized at one joint action profile $(action'_1, action'_2, ...)$, but that due to the historical probability distribution over action profiles, $action'_i$ does not maximize $E(utility, \mid action_i; observation_i)$, and therefore is not taken by agent i. In essence, there is a synchronization problem.

To simplify matters, we consider TG using the simplest possible RL algorithm, in which each learner makes no observations and uses no lookahead, choosing its action (ghost traffic level) at any moment by sampling a Gibbs distribution[1] over its estimates for the associated rewards [7; 23]. For stationary traffic patterns, with such RL algorithms the natural performance measure is given by the function taking joint-action profiles to the world reward.

One distributed approach to such a search problem that avoids the synchronization difficulty of TG while still only requiring the centralized broadcast of information that isn't personalized is simulated annealing (SA) [1; 9]. The variant of SA considered here repeats a two-step process. In the decentralized "exploration step", all agents make (usually small) random changes in their action. In the subsequent centralized "exploitation step", a Gibbs distribution over the associated rewards is sampled to choose between the new action profile and the previous one. That (non-personalized) choice — use your new action or your previous one — is then broadcast out to all the agents. (Note that such a broadcast is no more personalized than the world reward value that must be similarly broadcast in TG.) The major disadvantage of SA compared to distributed RL is that only one "learner" is exploited in SA, at the central location.

[1] I.e., choose action a_i with estimated reward R_i with probability $p(a_i) = \frac{\exp(-R_i/T)}{\sum_j \exp(-R_j/T)}$, where T determines the "temperature" parameter which controls the exploration/exploitation tradeoff.

In this article we introduce "bi-utility search", which is a hybrid of SA and distributed RL that combines the strengths of both. Bi-utility search is identical to SA, except that the random perturbation of each agent's action occurring in the exploration step is replaced by a perturbation determined by the RL algorithm of that agent. Since the exploitation of SA is used, the synchronization problem of distributed RL is avoided. On the other hand, since the exploration of distributed RL is used, the blind random nature of SA is replaced by a (hopefully) more intelligent RL-guided process. Particularly in large domains, we would expect that that guiding might provide major improvements in performance.

3 Constellations of Satellites

In our simulations each **satellite** s_i has a storage **capacity** c_i, measured in amount of data. It also has a **bandwidth** (or "link capacity", also measured in amount of data, implicitly per unit time interval) b_{ik} to each other satellite k. Earth is simply a special satellite with $c_0 = \infty$. At each time step t, an amount of **new data** y_{ijt} of importance j is introduced to the system at satellite i. We add y_{ijt} to the total amount of data of importance j received by i from all other satellites at t and then to the amount of unsent data on the disk left over from the previous time step to get the total **volume** of data of importance j at i at t, v_{ijt}. If $\sum_j v_{ijt} > c_i$, then that difference constitutes the amount of data lost at satellite i at t. We assume that the same proportion of data is dropped for each importance level, since once the disk is full the satellite is unable to examine any data sent to it and determine its priority.

Define l_{ijt} to be the amount of data of importance j **lost** (i.e., dropped) at satellite i at t. Define the associated cost as $l_{ijt} w_j$ for some **weights** w_j. Then the objective function we wish to maximize is the time-average of

$$G_t \equiv 1 - \frac{\sum_{ij} w_j l_{ijt}}{\sum_{ij} w_j y_{ijt}} . \quad (1)$$

Denote a path by a sequence of satellites $\vec{s} \equiv s_1, \ldots, s_p$, where s_1 represents the originating satellite and s_p is the satellite which ultimately sends the packet to Earth. At each t each satellite i evaluates a potential decision to send some data to satellite k by estimating the "headroom" of the optimal path to Earth beginning at k. The headroom of path $\vec{s}$ is the available room for additional data, given the available storage capacity on each satellite along $\vec{s}$ and the link capacity of each connection along $\vec{s}$. Formally, the headroom $H(\vec{s})$ of a path $\vec{s}$ is given by:

$$H(\vec{s}) = \min_{q \in \{1, \ldots p\}} (\min(b_{s_q, s_{q+1}}, c_{s_{q+1}} - \sum_j v_{s_{q+1}, j, t})) . \quad (2)$$

The presumption is that a path that currently has headroom should be favored over one with low headroom, since the likelihood of data being dropped is lower. This is just like how a path with low current cost is favored

over a path with high current cost in traditional Shortest Path (SP) data routing [2; 3]).

Note that in a real system, a particular satellite i does not have access to the precise $v_{jkt}, j \neq i$ at the current t. Hence the current headroom values would have to be estimated by satellites (just as current cost values are estimated in SP routing). Because we are interested in how to improve the performance of the base algorithm, in these experiments we ignore this issue and supply each satellite with the current v_{jkt}. (The estimation of the v_{jkt} would introduce a systematic error that should not affect the ranking of the algorithms discussed below).

It is straightforward to calculate the maximum headroom path from each satellite to Earth using a version of Dijkstra's shortest path algorithm [3; 10]. Let H_{ijt} be the time t headroom of the optimal path originating at satellite i and with the first hop being to satellite j. The satellite at which data originates does not directly decide on the full path to Earth taken by the data it transmits; it simply decides on the first hop in the path and sends its data to the appropriate satellite based on the H_{ijt}'s. (Similarly, in traditional data routing, a router only selects the first hop along the seemingly shortest path, based on the costs). More precisely, define

$$j^*_{it} \equiv \text{argmax}_j H_{ijt}. \quad (3)$$

If $H_{ij^*_{it}} > v_{ikt}$, then all of v_{ikt} $\forall k$ is sent to satellite j^*_{it} and $H_{ij^*_{it}}$ is updated by subtracting v_{ikt} off of the headroom estimate to reflect the fact that that much data has already been sent to that satellite. If $H_{ij^*_{it}} < v_{ikt}$, then an amount $H_{ij^*_{it}}$ is sent (highest importance data first) to j^*_{it} and $H_{ij^*_{it}}$ is updated to equal zero.

This procedure is then repeated until either:

1. $v_{ikt} = 0$ $\forall k$, or
2. $H_{ijt} = 0$ $\forall j$.

If the second condition occurs before all data has been routed, then the remaining data is not sent anywhere and instead kept on the disk until the next iteration in the hopes of routing it successfully then.

While this routing algorithm performs respectably, it is susceptible to the same phenomena that hamper traditional SP routing [20]: the satellites do not explicitly act to optimize G, and can therefore potentially work at cross-purposes. To alleviate this we introduce additive perturbations δ_{ijt} to the headroom estimates H_{ijt} and then perform the routing according to the perturbed headroom estimates. The δ_{ijt} are free parameters to be determined via an optimization algorithm, and because their effects on the headroom is the same as that of actual data, we call them "ghost" traffic. Our goal then is to find the $\{\delta_{ijt}\}$ at each t that optimize the average of $G_t(\{\delta_{ijt}\})$ — a search problem. In this paper we consider performing that search via two variants of simulated annealing, team game reinforcement learning and bi-utility search.

4 Experimental Results

In our experiments we had three importance levels ($w_j \in \{1, 2, 3\}$). For each satellite, for each importance level, at each t, new traffic was introduced by uniformly sampling [0.0, 0.5]. We used a network of moderate connectivity with 20 satellites altogether (150 δ's for each t). For the purposes of learning, we associated the actions of an "agent" at t with the setting of δ_{ijt} for a particular ij pair.

As mentioned before, to focus on how the individual RL-based agents' action decisions are exploited in the overall search algorithm rather than how those decisions are made in the first place, we used the simplest possible RL algorithms in the TG agents. Since they make no observations, formally each such an agent has one "state". Each agent's possible actions (ghost traffic on a particular link) are chosen from the set of discrete values {-5, -4, ..., 4}. To reflect the nonstationarity in the environment, each agent applies proportional updating to average its empirically collected training data, giving an estimated reward value for each candidate action. (We had the geometric updating factor set to .1 always.) The RL algorithm then decides what action to take by sampling a Gibbs distribution over those associated reward estimates (cf. Section 1), using a decaying temperature parameter. Temperatures annealed linearly from .1 down to .01 after 5000 τ intervals had passed.

We investigate five overall search algorithms. Each of them uses "time steps", labeled by τ, that consist of 200 consecutive t values of the underlying system. "G_τ" for each such "time step" is then defined to be the average G_t over that interval. (Note that successive values of G_τ are not statistically independent; some "leakage" of system behavior will occur across the boundary between successive τ.)

- **Baseline:** Algorithm outlined in Section 3; no learning.
- **Team Games (TG):** Each agent uses the RL algorithm described above to try to independently maximize G_τ at each τ.
- **Simulated Annealing (SA):** A two-step process is iterated. First a the δ vector is perturbed (uniformly) randomly and the resulting δ vector is implemented in the system. The ensuing G_τ is measured (exploration), and at the next τ that vector is probabilistically "accepted or rejected" in favor of the pre-perturbation vector, by sampling a Gibbs distribution over the two associated recorded G_τ values (exploitation). The vector so chosen is implemented, the ensuing G_τ measured, and the process repeats.
- **Localized Simulated Annealing:** Simulated annealing modified by restricting each component of the new perturbation vector to be at most one bin away from the old one.
- **G-Based Simulated Annealing (bi-utility search):** Localized simulated annealing modified

by having each component of the new perturbation vector set by having the associated TG agent choose among the 3 candidate bins.

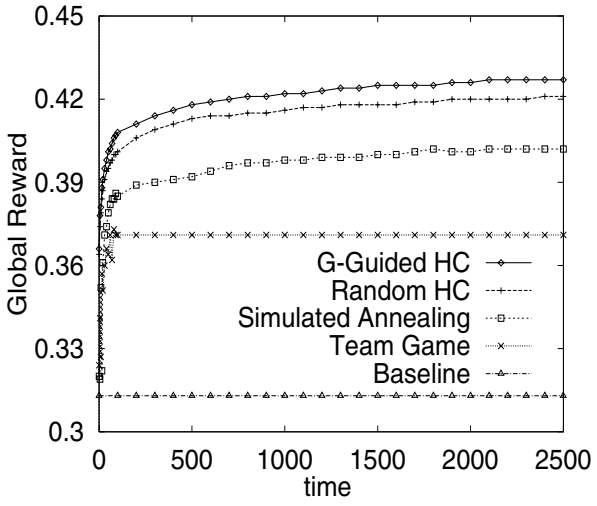


Figure 1: Overall performance of the algorithms in the communication satellites problem

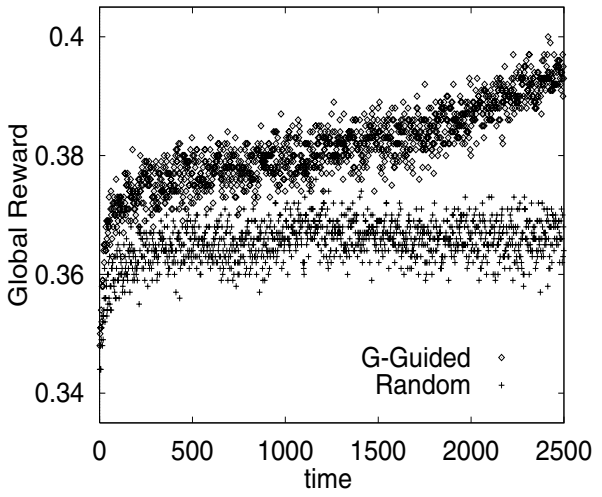


Figure 2: Exploration step performance for localized and G-guided simulated annealing.

Figure 1 compares the performance of these five algorithms on the network described above, averaged over 100 runs (the resulting error bars are too small to depict). The team game RL algorithm performs better than the baseline algorithm. However, the simulated annealing results show that TG is falling prey to problems like that of synchronization. When SA's minimal need for centralized broadcast information can be met, it should be used rather than TG. Next, note that the localized version of of SA significantly outperforms standard SA, reflecting the inherent smoothness the surface being searched. Finally, while both localized SA and bi-utility search are superior to both TG and SA, the G-guided version (bi-utility search) is the clear winner. This demonstrates that that algorithm does successfully combine the strengths of simulated annealing and reinforcement learning.

Figure 2 shows the G_τ of the δ's generated by the exploration steps of the two algorithms. RL-based exploration clearly produces better states, and this gap in performance increases in time. This translates into its higher performance.

5 Discussion

In search problems over large spaces, even if it is physically possible to use a centralized search algorithm, often one might expect that better performance would be achieved using a distributed rather than a centralized algorithm, due to its parallelization advantage. This raises the issue of what algorithms to run on each of the "agents" in such a distributed algorithm. Recent work has demonstrated that Reinforcement Learning (RL) can be quite gainfully applied to centralized search algorithms [6; 14; 26]. Here we investigate extending RL into a distributed setting.

We start by presenting a distributed RL search algorithm very similar to Team Games (TG) [8; 23]. In particular, we present experiments showing how to use this algorithm to determine the "ghost traffic" to be introduced into a pre-fixed algorithm designed for routing communication data across a constellation of satellites, in the hopes of fooling that algorithm into performing better. Our experiments indicate that this distributed RL algorithm significantly reduces the importance-weighted amount of data lost in such a domain.

Unfortunately, TG has many shortcomings, including a "signal-to-noise" problem and a "synchronization" problem. The first problem can be addressed using the COllective INtelligence (COIN) framework [20; 23; 24]. The second can be obviated if (non-personalized) periodic broadcast signals are allowed. In particular, Simulated Annealing (SA) only uses such signals, and avoids the synchronization problem. In fact, we find that SA outperforms TG in our problem domain.

We introduce bi-utility search as a way to exploit the parallelization advantage of a distributed algorithm without incurring the difficulties of TG. This procedure interleaves a distributed algorithm (e.g., TG) with SA, by using the distributed algorithm to govern an exploration step, which is followed by the exploitation step of SA. Our experiments show that TG-based bi-utility search substantially outperforms both TG and SA in our constellation of communication satellites domain.

As investigated here, bi-utility search uses Gibbs sampling both for the exploration and exploitation steps. It also uses world reward in both of those Gibbs samplers. None of that is *a priori* necessary; bi-utility search is a general approach for combining a centralized exploita-

tion step with a decentralized exploration step. In particular, the distributed algorithm can be based on the COIN framework, and thereby avoid the signal-to-noise problem of TG. Preliminary results not reported here indicate that such COIN-based bi-utility search performs even better than TG-based bi-utility search. We are currently investigating these issues further.

References

[1] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley and Sons, 1989.

[2] R. E. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

[3] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, Englewood Cliffs, NJ, 1992.

[4] K. D. Boese. *Models for Iterative Global Optimization*. PhD thesis, University of California, Los Angeles, 1996.

[5] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 16(2):101–113, 1994.

[6] J. A. Boyan and A. Moore. Learning evaluation functions for global optimization and boolean satisfiability. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. AAAI Press, 1998.

[7] C. Claus and C. Boutilier. The dynamics of reinforcement learning cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752, Madison, WI, June 1998.

[8] R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems - 8*, pages 1017–1023. MIT Press, 1996.

[9] R. Diekmann, R. Luling, and J. Simon. Problem independent distributed simulated annealing and its applications. In *Applied Simulated Annealing*, pages 17–44. Springer, 1993.

[10] E. Dijkstra. A note on two problems in connection with graphs. *Numeriche Mathematics*, 1(269-171), 1959.

[11] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242–250, June 1998.

[12] L. P. Kaelbing, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[13] S. Kirkpatrick, C. D. Jr Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, May 1983.

[14] M. L. Littman and J. Boyan. A distributed reinforcement learning scheme for network routing. In *Proceedings of the 1993 International Workshop on Applications of Neural Networks to Telecommunications*, pages 45–51, 1993.

[15] R. Moll, A. G. Barto, and T. J. Perkins. Learning instance-independent value functions to enhance local search. In *Advances in Neural Information Processing Systems - 11*, pages 1017–1023. MIT Press, 1999.

[16] D.T. Pham and D. Karaboga. *Intelligent Optimization Techniques*. Springer, London, UK, 2000.

[17] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, New York, 1999.

[18] T. Sandholm and R. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 37:147–166, 1995.

[19] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[20] K. Tumer and D. H. Wolpert. Collective intelligence and Braess' paradox. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 104–109, Austin, TX, 2000.

[21] R. V. V. Vidal, editor. *Applied Simulated Annealing (Lecture Notes in Economics and Mathematical Systems)*. Springer, 1993.

[22] D. H. Wolpert, S. Kirshner, C. J. Merz, and K. Tumer. Adaptivity in agent-based routing for data networks. In *Proceedings of the fourth International Conference of Autonomous Agents*, pages 396–403, 2000.

[23] D. H. Wolpert and K. Tumer. An Introduction to Collective Intelligence. Technical Report NASA-ARC-IC-99-63, NASA Ames Research Center, 1999. URL:`http://ic.arc.nasa.gov/ic/projects/coin_pubs.html`. To appear in Handbook of Agent Technology, Ed. J. M. Bradshaw, AAAI/MIT Press.

[24] D. H. Wolpert, K. Tumer, and J. Frank. Using collective intelligence to route internet traffic. In *Advances in Neural Information Processing Systems - 11*, pages 952–958. MIT Press, 1999.

[25] D. H. Wolpert, K. Wheeler, and K. Tumer. Collective intelligence for control of distributed dynamical systems. *Europhysics Letters*, 49(6), March 2000.

[26] W. Zhang and T. G. Dietterich. Solving combinatorial optimization tasks by reinforcement learning: A general methodology applied to resource-constrained scheduling. *Journal of Artificial Intelligence Reseach*, 2000.

Fast Concurrent Reinforcement Learners

Bikramjit Banerjee & Sandip Sen
Math & Computer Sciences Department
University of Tulsa
Tulsa, OK 74104
{*bikram, sandip*}*@euler.mcs.utulsa.edu*

Jing Peng
Dept. of Computer Science
Oklahoma State University
Stillwater, OK 74078
jpeng@cs.okstate.edu

Abstract

When several agents learn concurrently, the payoff received by an agent is dependent on the behavior of the other agents. As the other agents learn, the reward of one agent becomes non-stationary. This makes learning in multiagent systems more difficult than single-agent learning. A few methods, however, are known to guarantee convergence to equilibrium in the limit in such systems. In this paper we experimentally study one such technique, the minimax-Q, in a competitive domain and prove its equivalence with another well-known method for competitive domains. We study the rate of convergence of minimax-Q and investigate possible ways for increasing the same. We also present a variant of the algorithm, minimax-SARSA, and prove its convergence to minimax-Q values under appropriate conditions. Finally we show that this new algorithm performs better than simple minimax-Q in a general-sum domain as well.

1 Introduction

The reinforcement learning (RL) paradigm provides techniques using which an individual agent can optimize environmental payoff. However, the presence of a non-stationary environment, central to multi-agent learning, violates the assumptions underlying convergence proofs of single agent RL techniques. As a result, standard reinforcement learning techniques like Q-learning are not guaranteed to converge in a multi-agent environment. The focus of convergence for multiple, concurrent learners is on an equilibrium strategy-profile rather than optimal strategies for the individual agents.

The stochastic-game (or *Markov Games*) framework, a generalization of Markov Decision Processes for multiple controllers, has been used to model learning by agents in purely competitive domains [Littman, 1994]. In that work, the author has presented a minimax-Q learning rule and evaluated its performance experimentally. The Q-learning rule, being a model-free and online learning technique, is particularly suited for multi-stage games, and as such, an attractive candidate for multiagent-learning in uncertain environments. Hu and Wellman (1998) extended Littman's framework to enable the agents to converge to a mixed-strategy Nash equilibrium. There have also been other research on using the stochastic-game framework in multiagent learning, such as an empirical study of multiagent Q-learning in semi-competitive domains [Sandholm and Crites, 1996] among others.

The first two techniques mentioned above are known to converge in the limit. In this paper we show that these two techniques are essentially identical in purely competitive domains. To increase the convergence rate of the minimax-Q rule, we extend it to incorporate SARSA [Rummery, 1994; Sutton and Burto, 1998] and Q(λ) [Peng and Williams, 1996] techniques, and study the convergence rates of these different methods in a competitive soccer domain [Littman, 1994].

2 Multiagent Q-learning

Definition 1 *A Markov Decision Process (MDP) is a quadruple $\{S, A, T, R\}$, where S is the set of states, A is the set of actions, T is the transition function, $T : S \times A \rightarrow PD(S)$, PD being a probability distribution, and R is the reward function, $R : S \times A \rightarrow \Re$.*

A multiagent reinforcement-learning task can be looked upon as an extended MDP, with S specifying the joint-state of the agents, A being the joint-actions of the agents, $A_1 \times A_2 \times \ldots A_n$ where A_i is the set of actions available to the ith agent, T as the joint-transition function, and the reward function is redefined as $R : S \times A \rightarrow \Re^n$. The functions T and R are usually unknown. The goal of the ith agent is to find a strategy π_i that maximizes its expected sum of discounted rewards, $v(s, \pi_i) = \sum_{t=0}^{\infty} \gamma^t E(r_t^i | \pi_i, \pi_{-i}, s_0 = s)$ where s_0 is the initial joint-state, r_t^i is the reward of the ith agent at time t, $\gamma \in [0, 1)$ is the discount factor, and π_{-i} is the strategy-profile of i's opponents.

Definition 2 *A bimatrix game is given by a pair of matrices, (M_1, M_2), (each of size $|A_1| \times |A_2|$ for a two-agent game) where the payoff of the ith agent for the joint action (a_1, a_2) is given by the entry $M_k(a_1, a_2)$, $\forall (a_1, a_2) \in A_1 \times A_2$, $k = 1, 2$.*

Each stage of an extended-MDP for two agents (it can be extended to n agents using n-dimensional tables instead of matrices) can be looked upon as a bimatrix game. A *zero-sum game* is a special bimatrix game where $M_1(a_1, a_2) + M_2(a_1, a_2) = 0$, $\forall (a_1, a_2) \in A_1 \times A_2$.

Definition 3 *A mixed-strategy Nash Equilibrium for a bimatrix game (M_1, M_2) is a pair of probability vectors (π_1^*, π_2^*) such that*

$$\pi_1^{*T} M_1 \pi_2^* \geq \pi_1^T M_1 \pi_2^* \ \ \forall \pi_1 \in PD(A_1).$$

$$\pi_1^{*T} M_2 \pi_2^* \geq \pi_1^{*T} M_2 \pi_2 \ \ \forall \pi_2 \in PD(A_2).$$

where $PD(A_i)$ is the set of probability-distributions over the ith agent's action space.

No player in this game has any incentive for unilateral deviation from the Nash equilibrium strategy, given the other's strategy. There always exists at least one such equilibrium profile for an arbitrary finite bimatrix game [Nash, 1951].

An individual learner may, but need not, use a model of the environment to learn the transition and reward functions. Q-learning is one example of model-free learning. In greedy policy Q-learning, an agent starts with arbitrary initial Q-values (or action values) for each state-action pair (s, a), and repeatedly chooses actions, noting its rewards and transitions and updating Q as

$$Q^{t+1}(s_t, a_t) = (1 - \alpha_t) Q^t(s_t, a_t) + \alpha_t [r_t + \gamma v^t(s_{t+1})]$$

where

$$v^t(s_{t+1}) = \max_a Q^t(s_{t+1}, a). \tag{1}$$

and $\alpha_t \in [0, 1)$ is the learning rate. Watkins and Dayan (1992) have proved that the above iteration converges to optimal action values under infinite sampling of each state-action pair and a particular schedule of the learning rate.

In the case of multiagent learning, the above iteration would not work, since the maximization over one's action is insufficient in the presence of multiple actors. However, if the reward-function of the opponent is negatively correlated, then actions can be selected by solving the bimatrix-game $(M(s), -M(s))$ greedily for the opponent, and pessimistically for oneself, to guarantee a minimum expected payoff. This produces Littman's minimax-Q algorithm for simultaneous-move zero-sum games, for which the value-function for agent 1 is

$$v_1^t(s_{t+1}) = \max_{\pi \in PD(A)} \min_{o \in O} \pi^T Q_1^t(s_{t+1}, ., o), \tag{2}$$

where A and O are the action-sets of the learning-agent (agent 1) and its opponent respectively, and $Q_1^t(s_{t+1}, ., o)$ is a vector of action values of the learner corresponding to its opponent's action o. The current policy $\pi(s)$ can be solved by linear-programming for the constrained, minimax optimization on $Q(s, ., .)$. The minimax-Q learning rule has been proved to converge to optimal action values [Szepesvári and Littman, 1997].

For general-sum games, however, the ith agent needs to know π_{-i}, in absence of which it has to model its opponents. In such games, each agent can observe the other agent's actions and rewards and maintains separate Q-tables for each of them in addition to its own [Hu and Wellman, 1998].

The value-function for agent 1 in this case is

$$v_1^t(s_{t+1}) = \pi_1^*(s_{t+1})^T Q_1^t(s_{t+1}, ., .) \pi_2^*(s_{t+1}), \tag{3}$$

where (π_1^*, π_2^*) are the Nash-strategies of the agents for the bimatrix game $\{Q_1^t(s_{t+1}, ., .), Q_2^t(s_{t+1}, ., .)\}$, which can be solved by quadratic-programming technique [Mangasarian and Stone, 1964]. In zero-sum games, the value-function in (3) simplifies to

$$v_1^t(s_{t+1}) = \max_{\pi_1 \in PD(A_1)} \min_{\pi_2 \in PD(A_2)} \pi_1^T Q_1^t(s_{t+1}, ., .) \pi_2. \tag{4}$$

This algorithm converges to a Nash equilibrium, for a restrictive class of Nash-equilibria [Hu and Wellman, 1998], that in addition to the constraints imposed by its definition, satisfies the following

$$\pi_1^{*T} Q_1^t(s_{t+1}) \pi_2^* \leq \pi_1^{*T} Q_1^t(s_{t+1}) \pi_2 \ , \forall \pi_2 \in PD(A_2),$$
$$\pi_1^{*T} Q_2^t(s_{t+1}) \pi_2^* \leq \pi_1^T Q_2^t(s_{t+1}) \pi_2^* \ , \forall \pi_1 \in PD(A_1).$$

Though this may not be true in general, it holds for zero-sum domains, where any deviation by the opponent from its equilibrium-strategy decreases its expected payoff, thus increasing the modeler's expected payoff. Hence, the convergence of this algorithm is guaranteed in zero-sum domains. Furthermore, the algorithms developed by Littman (1994) and Hu and Wellman (1998) (we call the latter Nash-Q) differ structurally only in the use of rules (2) and (4) respectively, in updating Q. But contrary to the statement made by Hu and Wellman (1998), these expressions are functionally equivalent in zero-sum games. Game theory [Thie, 1998] states that in a zero-sum game

$$\max_{\pi_1 \in PD(A_1)} \min_{\pi_2 \in PD(A_2)} \pi_1^T Q_1(s, ., .) \pi_2$$
$$= \max_{\pi_1 \in PD(A_1)} \min_{o \in A_2} \pi_1^T Q_1(s, ., o), \ \forall s \in S. \tag{5}$$

An informal argument may go as follows. Let $Q_1(s, ., .) = (q_1, q_2, \cdots, q_n)$, where q_i represents the ith column of $Q_1(s, ., .)$, and $n = |A_2|$. Then $\pi_1^T Q_1(s, ., .) = (\pi_1^T q_1, \pi_1^T q_2, \cdots, \pi_1^T q_n)$ for any $\pi_1 \in PD(A_1)$. When $\pi_1^T q$ is treated as a random variable from the distribution π_2, we have

$$\begin{aligned} \min_{\pi_2 \in PD(A_2)} \pi_1^T Q_1(s, ., .) \pi_2 &= \min_{\pi_2 \in PD(A_2)} E_{\pi_2}[\pi_1^T q] \\ &\geq \min_{o \in A_2} \pi_1^T Q_1(s, ., o). \end{aligned}$$

On the other hand, since any pure strategy is also a mixed strategy, we have

$$\begin{aligned} \min_{\pi_2 \in PD(A_2)} \pi_1^T Q_1(s, ., .) \pi_2 &= \min_{\pi_2 \in PD(A_2)} E_{\pi_2}[\pi_1^T q] \\ &\leq \min_{o \in A_2} \pi_1^T Q_1(s, ., o). \end{aligned}$$

Consequently we have the equality (5). Therefore we observe that both minimax-Q and Nash-Q compute identical policies and value-functions in zero-sum domains.

3 Expediting minimax-Q learning

Since minimax-Q and Nash-Q algorithms are equivalent in the purely competitive domains that we consider in this paper, we focus on the minimax-Q algorithm since it does not require maintenance of opponents' Q functions and hence is resource-efficient. We now turn to explore possible ways to speed-up the chosen algorithm.

3.1 Fast minimax-Q(λ)

Since Q-learning updates only the last state with the reinforcements, it is substantially slow in updating the action values. A well-known technique to speed-up single-agent Q-learning is to integrate it with TD(λ) reward-estimation scheme, producing the Q(λ)-learning rule [Peng and Williams, 1996]. For experimentation, we have used a faster version of the Peng-Williams' algorithm, where the Q updates are 'lazily' postponed until necessary [Wiering and Schmidhuber, 1998]. Q(λ) can be applied to each of the two learning schemes in the previous section, by defining $v(s_{t+1})$ in the Q(λ) algorithm by the equation in (2) or (3) as the case may be. The guarantee of convergence, however, may no longer hold.

3.2 Minimax-SARSA learning

The previous techniques were all off-policy learning rules, where the expected value of the next state is used to update the value of the current state. The SARSA(0) technique is, on the other hand, an on-policy learning rule that depends heavily on the actual learning policy followed [Rummery, 1994; Sutton and Burto, 1998]. In general, off-policy algorithms can separate control from exploration while on-policy reinforcement learning algorithms cannot. Despite this, on-policy algorithms with function approximation in single agent learning appear to be superior to off-policy algorithms in control as well as prediction problems [Boyan and Moore, 1995; Sutton, 1996; Tsitsiklis and Roy, 1997]. On-policy algorithms can learn to behave consistently with exploration [Sutton and Burto, 1998] Moreover, on-policy algorithms are more natural to combine with eligibility traces than off-policy algorithms are. This raises the following question that this research effort endeavors to answer: Can on-policy RL algorithms perform equally well in multiagent domains? In that case we can possibly achieve faster convergence of a hybrid of the SARSA technique and minimax-Q rule.

In a simple Q-learning scenario, the SARSA technique modifies the update rule (1) as $v^t(s_{t+1}) = Q^t(s_{t+1}, a_{t+1})$. Thus a SARSA rule learns the values of its own actions, and can converge to optimal values only if the learning policy chooses optimal actions in the limit. In a multiagent minimax-Q setting, the rule (2) would be replaced (for agent 1) by

$$v_1^t(s_{t+1}) = Q_1^t(s_{t+1}, a_{t+1}, o_{t+1}),$$

while the policy to choose actions would still be computed by the original minimax-Q rule. To achieve convergence of this rule to minimax-Q values, we follow an ϵ-minimax strategy that satisfies the need of infinite exploration while being minimax in the limit, i.e., ϵ decays to 0 in the limit. We call such exploration 'Minimax in the limit with infinite exploration' or MLIE. Our convergence result rests on the following lemma established by Singh *et al.* (2000).

Lemma 1 *Consider a stochastic process (α_t, Δ^t, F^t), $t \geq 0$, where $\alpha_t, \Delta^t, F^t : X \rightarrow \Re$ satisfy the equations*

$$\Delta^{t+1}(x) = (1 - \alpha_t(x))\Delta^t(x) + \alpha_t(x)F^t(x),$$

where $x \in X$, $t = 0, 1, 2, \ldots$. Let P_t be a sequence of increasing σ-fields such that α_0 and Δ^0 are P_0 measurable and α_t, Δ^t and F^{t-1} are P_t measurable, $t = 1, 2, \ldots$. Assume that the following hold:
1. X is finite.
2. $0 \leq \alpha_t(x) \leq 1$, $\sum_t \alpha_t(x) = \infty$, $\sum_t \alpha_t^2(x) < \infty$ w.p.1.
3. $\|E\{F^t(.)|P_t\}\|_W \leq \delta\|\Delta^t\|_W$[1] $+ c_t$, where $\delta \in [0, 1)$ and c_t converges to 0 w.p.1.
4. $Var\{F^t(x)|P_t\} \leq \beta(1 + \|\Delta\|_W)^2$, for some constant β.
Then, Δ^t converges to 0 with probability 1 (w.p.1).

The update rule for SARSA for agent 1 say, is

$$\begin{aligned} Q_1^{t+1}(s_t, a_t, o_t) &= (1 - \alpha_t)Q_1^t(s_t, a_t, o_t) + \\ &\quad \alpha_t[r_t^1 + \gamma Q_1^t(s_{t+1}, a_{t+1}, o_{t+1})]. \end{aligned} \quad (6)$$

We also note that the fixed point of the minimax-Q rule [Szepesvári and Littman, 1997] (for agent 1) is

$$\begin{aligned} Q_1^*(s_t, a_t, o_t) &= R_1(s_t, a_t, o_t) + \\ &\quad E_y[\max_{\pi_1} \min_o \pi_1^T Q_1^*(y, ., o)]. \end{aligned} \quad (7)$$

Now we state and prove the theorem for convergence of minimax-SARSA learning using Lemma 1.

Theorem 1 *The learning rule specified in (6) converges to the values in equation (7) with probability 1 provided a_t is chosen using an MLIE scheme at each step t, the immediate rewards are bounded and have finite variance, the Q-values are stored in lookup tables, the learning rate, α_t, satisfies condition 2 in Lemma 1, and the opponent plays greedily in the limit.*

Proof: (Outline) Writing x in Lemma 1 as (s_t, a_t, o_t) and Δ^t as $Q_1^t(s_t, a_t, o_t) - Q_1^*(s_t, a_t, o_t)$, and defining $\alpha_t(s, a, o) = 0$ unless $(s, a, o) = (s_t, a_t, o_t)$ $\forall t$, we have

$$\begin{aligned} F^t(s_t, a_t, o_t) &= r_t^1 + \gamma \max_{\pi_1} \min_o \pi_1^T Q_1^t(s_{t+1}, ., o) \\ &\quad - Q_1^*(s_t, a_t, o_t) + \gamma Q_1^t(s_{t+1}, a_{t+1}, o_{t+1}) \\ &\quad - \gamma \max_{\pi_1} \min_o \pi_1^T Q_1^t(s_{t+1}, ., o)], \end{aligned} \quad (8)$$

which gives rise to

$$F^t(s_t, a_t, o_t) = F_M^t(s_t, a_t, o_t) + \gamma[d_t(s_t, a_t, o_t)].$$

It can be shown that the measurability and variance conditions are satisfied and that

$$\|E\{F_M^t(., ., .)|P_t\}\| \leq \gamma_M \|\Delta^t\|$$

for some $\gamma_M \in [0, 1)$ (since minimax-Q operator is a contraction), according to the outline provided by Singh *et al.* (2000). The remaining task is to show that $\|E\{d_t(., ., .)|P_t\}\|$ vanishes in the limit, under MLIE exploration. We consider the following cases:

Case 1: $Q_1^t(s_{t+1}, a_{t+1}, o_{t+1}) \geq \max_{\pi_1} \min_o \pi_1^T Q_1^t(s_{t+1}, ., o)$. Since $\max_{\pi_1} \min_o \pi_1^T Q_1^t(s_{t+1}, ., o) \geq \min_o Q_1^t(s_{t+1}, a_{t+1}, o)$, we have $d_t(s_t, a_t, o_t) = |d_t(s_t, a_t, o_t)| \leq Q_1^t(s_{t+1}, a_{t+1}, o_{t+1}) - \min_o Q_1^t(s_{t+1}, a_{t+1}, o)$ and the corresponding expected value vanishes in the limit if the opponent plays greedily in the limit.

[1] for our purpose $\|.\|_W$ is a max-norm for a uniform weight-vector, W.

Case 2: $\max_{\pi_1} \min_o \pi_1^T Q_1^t(s_{t+1}, ., o) \geq Q_1^t(s_{t+1}, a_{t+1}, o_{t+1})$.

Again, $\max_{\pi_1} \min_o \pi_1^T Q_1^t(s_{t+1}, ., o) \leq \pi_1^{*T} Q_1^t(s_{t+1}, ., o_{t+1})$ where $\pi_1^* = \arg\max_{\pi_1} \min_o \pi_1^T Q_1^t(s_{t+1}, ., o)$. Hence $-d_t(s_t, a_t, o_t) = |d_t(s_t, a_t, o_t)| \leq \pi_1^{*T} Q_1^t(s_{t+1}, ., o_{t+1}) - Q_1^t(s_{t+1}, a_{t+1}, o_{t+1})$. The associated expected value vanishes again in the limit due to the assumption of an MLIE policy on part of agent 1, and independent of the opponent's behavior.

Let $C_t(s_t, a_t, o_t)$ be the maximum of the two upper limits on $|d_t(s_t, a_t, o_t)|$ established above. We see that $E\{|d_t(s_t, a_t, o_t)|\} \leq E\{C_t(s_t, a_t, o_t)\}$ and the r.h.s vanishes for each state-action tuple. Hence, $E\{|d_t(s_t, a_t, o_t)|\}$ vanishes for each state-action tuple, which implies that $\|E\{d_t(.,.,.)|P_t\}\|$ vanishes in the limit under MLIE exploration and optimal play by the opponent in the limit. Setting c_t in Lemma 1 to $\|E\{d_t(.,.,.)|P_t\}\|$, we conclude that minimax-SARSA rule converges to the minimax-Q values under MLIE exploration with probability 1, if the opponent plays greedily in the limit, and under appropriate structure of α_t. [Q.E.D.]

Note that **Case 1** needs the boundedness of Q_1^t that follows easily under additional assumptions. It might be argued that the condition of greedy play by the opponent in the limit is restrictive. However, this is typical of a convergence proof of on-policy algorithms that requires more details of the actions taken by the agents. Gains from on-policy algorithms in terms of learning efficiency and cost offset the condition of greedy play in the limit.

4 Experiments in a Competitive Domain

To evaluate the proposed schemes, we used the purely competitive soccer domain [Littman, 1994]. It is a 4×5 grid containing two agents, A and B, as shown in figure 1, that always occupy distinct squares. The goal of agent A is on the left, and that of B on right. The Figure 1 shows the initial positions of the agents, with the ball being given to an agent at random at the start of each game (agent B in figure). Each agent can choose from a fixed set of five actions at each state: going up, left, down or right, or staying where it is.

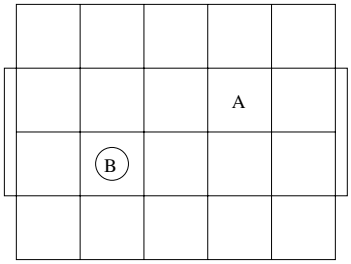


Figure 1: The experimental soccer domain.

When both the agents have selected their actions, these actions are executed in a random order. If an agent bumps onto another, the stationary agent receives the ball, and the movement fails. An agent receives reinforcements of +1 for a goal (or a sameside by the opponent) and -1 for a self-goal (or a goal by the opponent) to maintain the zero-sum character of the game, and in all other cases the reinforcement is zero. Whenever a non-zero reward is received, the game resets to the initial configuration. We shall call an agent following Littman's minimax-Q algorithm an M-agent.

In the training phase of the experiments, we performed symmetric training between two ordinary M-agents, two M-agents both using the $Q(\lambda)$ rule, and two M-agents both using the SARSA rule. The respective policies learnt, are denoted as MM_i, λMM_i, sMM_i, which are recorded at the end of each $i \times 10000$ iterations. Each training lasted 100,000 iterations in all. We used identical exploration-probabilities as that by Littman (1994) and the decay-factor for the learning-rate was set to 0.999954.

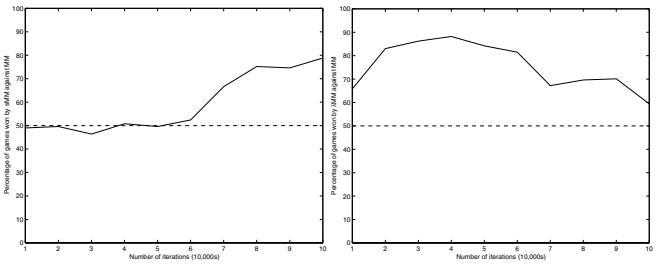


Figure 2: Games are played by sMM_i (left) and λMM_i (right) against MM_i for various values of i (horizontal axis). The percentages of wins (vertical axis) by the former in each case are plotted (averaged over 10 runs).

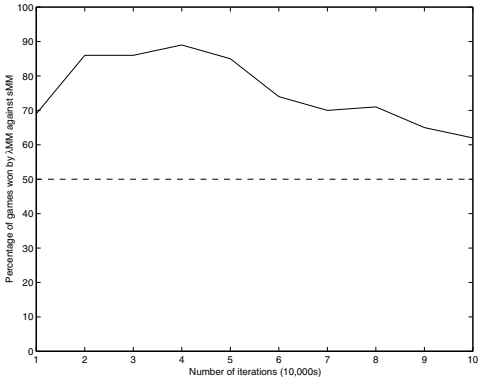


Figure 3: Games are played by λMM_i against sMM_i for various values of i. The percentages of wins (averaged over 10 runs) by the former in each case are plotted.

In the test phase, we allowed an sMM_i policy to play against an MM_i policy, for $i = 1 \ldots 10$. Each test iteration results in a draw with a probability of 0.05, to break possible deadlocks. The resultant percentages of win by the sMM_i policies over its opponent are reported in Figure 2 (left). The approximate trend suggests that ordinary minimax-Q initially dominates but minimax-SARSA gradually catches up and outperforms the former. In Figure 2 (right), the corresponding results from playing λMM_i against MM_i are shown. In

this case the minimax-Q(λ) rule outperforms the ordinary minimax-Q algorithm from the very beginning. However, the λMM policies gradually lose their edge as the ordinary minimax-Q rule learns better progressively. The figure 3 corroborates these observations, as λMM_i performs well against sMM_i, but this performance decays with increasing i. λ was set to 0.7 in both experiments.

We note that percentage of wins in such games may not be a good comparative estimator of the policies. A better estimator would be the average RMS deviations of the Q-values from their convergence values. However, the latter can be calculated in this domain only with extensive off-line computation. We also stress that the results reported are far from convergence, at which all the algorithms should perform equally well. The reason why sMM beats MM can be understood in the context of Q updates. While sMM uses the actual action value from the next state to update the current state, MM still uses the minimax value from the next state, which postpones relying on the individual table-entries. As a result, we expect MM to catch up (in Fig. 2 left) when learning continues.

5 Experiments in a General-sum Domain

We note that minimax-Q rule is applicable in general-sum domains as well, where the rationale of the assumption of minimizing policy of the opponent is to guarantee a minimum security level to the learner, instead of maximizing the reward of the opponent itself as in the zero-sum interpretation. The SARSA and $Q(\lambda)$ versions will still work in such domains. For the purpose of experimentation, we introduce a general-sum domain that we call "tightly coupled navigation." This is a 4×3 gridworld as shown in figure 4. The values in the lower left corner of each cell in figure 4 is the reward to agent 1 for reaching the state corresponding to that cell. Similarly the values in the upper right corner are those for agent 2. The rewards in this domain are state-based, i.e. the reward corresponding to a cell is received if the agents reach or remain in that cell. Here the agents are tightly coupled as they must always occupy the same cell. Each agent has three available actions in each state, viz. up, down, right. However, since they are coupled, they can move only when they choose the same action; otherwise they remain in the same state. The starting and the absorbing states have been shown in the figure 4. When the agents reach the goal state, each receives the reward 20 and without making any update in this iteration, the game restarts with the agents reshifted to the start-state and updates begin once again.

A more realistic scenario for this domain is car-driving. Consider two agents in the same car and each having a steering wheel in its hands. The car moves in a given direction if both move the wheels in that direction; otherwise the car does not move. There may be different paths that the agents wish to follow to reach their common goal. However, since they are tightly coupled, they must strike a compromise and find an intermediate path that both can be maximally satisfied with, given the coupling.

We have symmetrically trained two minimax-Q and two minimax-SARSA agents in this domain. The exploration probabilities for the agents in each iteration were the same as in the

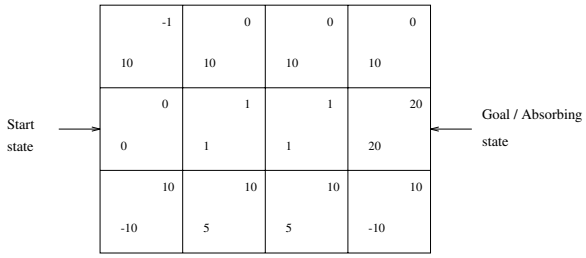


Figure 4: The tightly coupled navigation domain.

soccer domain, viz. 0.2. We varied the probability of reward-generation in each iteration using three values, viz. 0, 0.5 and 1.0, where 0 stands for the case where rewards are generated only when the agents reach the goal state. We wanted to study the effect of infrequent rewards, which is a realistic scenario in most practical domains, on the convergence of our algorithms. We expected the convergence rates to fall with more and more infrequent rewards. In order to study the convergence, the *exact* minimax-Q tables were computed off-line and an average RMS deviation of the learned Q-tables every 1000 training-iterations were plotted. The trainings lasted a total of 10,000 iterations.

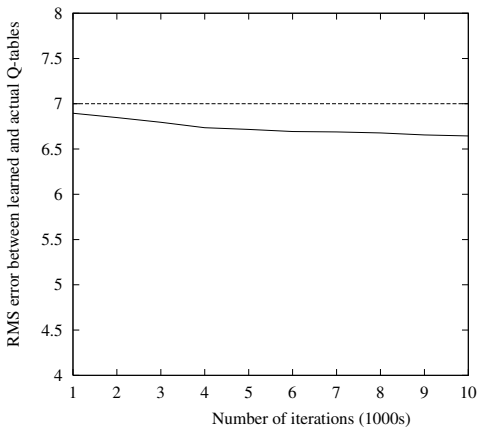


Figure 5: Mean RMS deviation plots of minimax-SARSA (solid) and ordinary minimax-Q for probability of reward-generation = 0.

From figures 5, 6 and 7, we can see that the minimax-SARSA rule always performs better than the ordinary minimax-Q rule. The errors in all the cases decrease monotonically which suggests that both the rules will eventually converge. As expected, the error-levels fall with increasing probability of reward-generation. A scrutiny of the minimax-Q tables show that the minimax path learned by each agent should be different from the Nash-equilibrium path that is corroborated by the learned Q-tables.

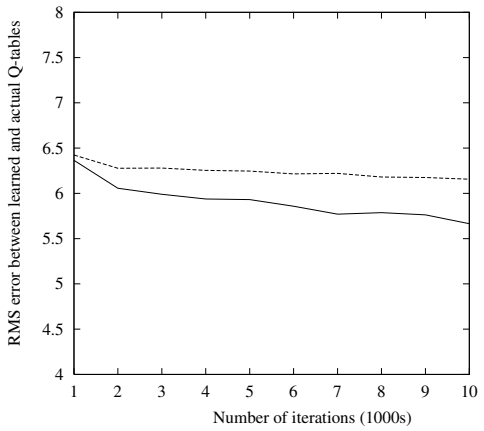


Figure 6: Mean RMS deviation plots of minimax-SARSA (solid) and ordinary minimax-Q for probability of reward-generation = 0.5.

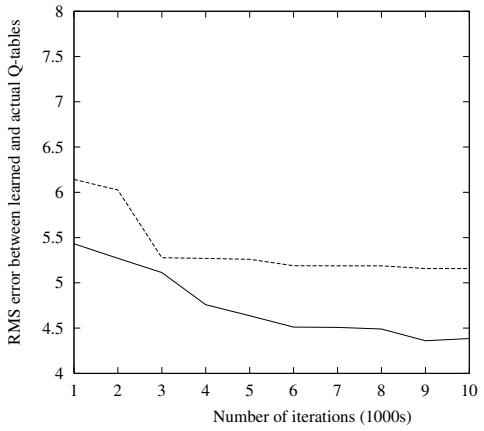


Figure 7: Mean RMS deviation plots of minimax-SARSA (solid) and ordinary minimax-Q for probability of reward-generation = 1.0.

6 Conclusion and Future work

We conclude that both the SARSA and $Q(\lambda)$ versions of minimax-Q learning achieve speed-up on Littman's minimax-Q rule, and more so for the $Q(\lambda)$ rule. Though this latter rule works well, we are not aware of the theoretical convergence properties of this method. Exploring these properties is one open area. We also note that a combination of minimax-SARSA and $Q(\lambda)$ to form what could be called minimax-SARSA(λ), would probably be more expedient than either of the two, by naturally combining their disjoint areas of expedience, seen in the plots in figure 2. Results from associated experiments are awaited. We could also substitute Nash-learning for minimax-learning and achieve Nash-Q(λ) and Nash-SARSA, specialized fast learning procedures for general-sum domains. A theoretical proof of convergence of such a Nash-SARSA would be along the same lines as presented in this paper for minimax-SARSA. We plan to conduct experiments with all these hybrid algorithms.

References

[Boyan and Moore, 1995] J.A. Boyan and A.W. Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems 7*, pages 369–376, 1995.

[Hu and Wellman, 1998] J. Hu and M. P. Wellman. Multi-agent reinforcement learning: Theoretical framework and an algorithm. In *Proc. of the 15th Int. Conf. on Machine Learning (ML'98)*, pages 242–250, San Francisco, CA, 1998. Morgan Kaufmann.

[Littman, 1994] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. of the 11th Int. Conf. on Machine Learning*, pages 157–163, San Mateo, CA, 1994. Morgan Kaufmann.

[Mangasarian and Stone, 1964] O. L. Mangasarian and H. Stone. Two-person nonzero-sum games and quadratic programming. *Journal of Mathematical Analysis and Applications*, 9:348 – 355, 1964.

[Nash, 1951] John F. Nash. Non-cooperative games. *Annals of Mathematics*, 54:286 – 295, 1951.

[Peng and Williams, 1996] J. Peng and R. Williams. Incremental multi-step Q-learning. *Machine Learning*, 22:283 – 290, 1996.

[Rummery, 1994] G. A. Rummery. *Problem solving with reinforcement learning*. PhD thesis, Cambridge University Engineering Department, 1994.

[Sandholm and Crites, 1996] T. Sandholm and R. Crites. On multiagent Q-learning in a semi-competitive domain. In G. Weiß and S. Sen, editors, *Adaptation and Learning in Multi-Agent Systems*, pages 191–205. Springer-Verlag, 1996.

[Sutton and Burto, 1998] R. Sutton and A. G. Burto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[Sutton, 1996] R. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, 1996.

[Szepesvári and Littman, 1997] Csaba Szepesvári and M.L. Littman. A unified analysis of value-function-based reinforcement-learning algorithms. In *Neural Computation*, 1997. submitted.

[Thie, 1998] P. R. Thie. *An Introduction to Linear Programming and Game Theory*. John Wiley and Son, 2nd. edition, 1998.

[Tsitsiklis and Roy, 1997] J.N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 1997.

[Wiering and Schmidhuber, 1998] M. Wiering and J. Schmidhuber. Fast online Q(λ). *Machine Learning, 33(1)*, pages 105–116, 1998.

MACHINE LEARNING AND DATA MINING

REINFORCEMENT LEARNING/ROBOTICS

Multi-Agent Systems by Incremental Gradient Reinforcement Learning

Alain Dutech **Olivier Buffet** **François Charpillet**
{buffet,dutech,charp}@loria.fr
LORIA, BP 239
54506 Vandoeuvre-lès-Nancy
France

Abstract

A new reinforcement learning (RL) methodology is proposed to design multi-agent systems. In the realistic setting of situated agents with local perception, the task of automatically building a coordinated system is of crucial importance. We use simple reactive agents which learn their own behavior in a decentralized way. To cope with the difficulties inherent to RL used in that framework, we have developed an incremental learning algorithm where agents face more and more complex tasks. We illustrate this general framework on a computer experiment where agents *have* to coordinate to reach a global goal.

1 Introduction

Multi-Agent Systems (MAS) - systems where autonomous entities called agents interact with each other - are a growing interest in the artificial intelligence community, both in terms of research and applications (see [Ferber, 1999]). Whereas MAS are usually build "by hand" using simulations to tune up the system to reach a desired global behavior, this paper deals with a key problem : the design of MAS in an automated fashion *through learning*.

Learning in MAS can indeed take many forms [Stone and Veloso, 2000]. We opted for Reinforcement Learning (RL) methods [Sutton and Barto, 1998] as they do not require a teacher knowing before hand a solution of the problem. An evaluation of the current agents' actions, using a scalar value for example, is enough to learn.

Furthermore, we consider Multi-Agent Systems composed of simple reactive situated agents with local perceptions. Thus, the system is easier to build and can nevertheless solve complex problems as, in this **decentralized** framework, each agent faces simpler tasks.

As RL suffers from combinatorial explosion, decentralization of the learning process should bring the same benefits, i.e. each agent learns its own behavior by itself which simplifies the task to be learned. Besides, it is consistent with the localized aspect of realistic situated agents. But, in this context, two major difficulties burden RL :

- **hidden global state.** Usual situated agents can only rely on an imperfect, local and partial perception of their world. Then, the global state of the system stays unknown, which prevents classical RL algorithms from finding an optimal policy, as shown by [Singh *et al.*, 1994].
- **credit assignment problem.** When a positive reward is given by the environment, it is not always evident to credit positively the "good" actions that led to this reward. With many agents, this problem is even more crucial as we must also decide which agents to reward.

Our answer to these problems is a decentralized **incremental** learning algorithm based on a classical RL technique to find **stochastic** policies. Actually, agents with stochastic behaviors are more adapted to the problem of partial perception. By *incremental*, we mean that agents are progressively pitted against harder and harder tasks so as to progressively learn a complex behavior. Another aspect of the incremental learning is to begin learning with very few agents, so as to minimize coordination and cross-work actions, and then export these basic behaviors to tasks with more and more agents. Eventually, behaviors can be further refined by learning in these more demanding environments. Thus, the originality of our approach is twofold : learning is decentralized and incremental.

In this paper, we present our method for learning in a multi-agent system and an experiment on which we tested our ideas. Section 2 gives the details of our framework, then, Sections 3, 4 and 5 describe the experiments conducted to test the viability of our approach. A discussion about our work follows in Section 6, highlighted by some comparisons from other similar works. Future directions and conclusive remarks end the paper in Section 7.

2 Our Framework

In this part, we first present the kind of agents we consider. Then we show the problems one faces when using reinforcement learning in a multi-agent setting. Lastly, we explain how incremental learning brings a partial solution to these problems.

2.1 The agents

We are interested in designing Multi-Agent Systems (MAS) by having each individual agent learn its behavior. As written earlier, we have decided to work with very simple reactive

agents for complexity reasons. Besides, it allows us to concentrate on the learning aspect of the design. Among many possible choices, our agents can be characterized as :

- **situated with local perception** : Local perceptions are more thoroughly discussed in Section 2.4.
- **possibly heterogeneous** : through the learning process where agents learn individually, each agent can acquire a different behavior from the others.
- **cooperative** : all agents share the same goal and they *will* have to coordinate to reach it.

We say nothing about other characteristics (for example communications) as they are not important for our framework and could be easily incorporated into the agents.

2.2 Limitations of classical RL

Reinforcement Learning (RL) methods are very appealing ways of learning optimal memoryless behaviors for agents as they only require a scalar feedback from the system to the agents for them to learn. Besides, these techniques can be used when there is uncertainty in the world's evolution.

But the convergence of RL algorithms (like *Q-Learning* or *TD(λ)*) has only been proven for Markov Decision Processes (MDP). A MDP is defined as a $< \mathcal{S}, \mathcal{A}, T, r >$ tuple, $\mathcal{S}$ being a finite set of states and $\mathcal{A}$ a finite set of actions. When the system is in given state s, an action a being chosen, the probability for the system to end in state s' is given by $T(s, a, s')$. After each transition, the environment generates a reward given by $r(s, a)$. The problem is then to find the optimal mapping $\pi(s, a)$ between states and actions so as to maximize the reward received over time, usually expressed as a utility function $V(s) = \sum_{t=0}^{\infty} \gamma^t (s_t | s_0 = s)$. Such a mapping is called a policy.

As pointed out by [Boutilier, 1996] the evolution of the kind of MAS we are interested in is a MDP. As such, it could be solved using classical reinforcement learning algorithms where the state of the system is the composition of the states of all agents and an action is a joint action composed of all individual actions of the agents. Thus, the number of states and actions in a centralized view of the problem should quickly prove to be too big for RL to be applied, as illustrated in Section 3.1. Besides, solving our problem this way would mean to solve it in a centralized way whereas we aim at a non-centralized solution as each agent should learn by itself.

Unfortunately, as shown by [Bernstein *et al.*, 2000], solving the problem in a non-centralized way when the agents only have a partial perception of the system's state is NEXP-complete, i.e. there is provably no polynomial algorithm to solve the problem. We face two major difficulties :

1. **Non-stationary transitions.** Reactive agents with a local view of the environment can not use joint actions to solve the problem. In fact, other agents are unpredictable elements of the environment. As a consequence, the transitions from one state of the system to another, as seen by an agent, are non-stationary : for example the probability for an agent to move ahead depends greatly on the actions of other adjacent agents.
2. **Partial observability.** As the agent's perception is local they can not know the global state of the problem. As such, the problem at hand belongs to the class of *partially observed* Markov decision models (see [Littman *et al.*, 1995]).

Classical stationary Partially Observed Markov Decision Processes are nearly impossible to solve when there are more than a hundred states [Dutech, 2000]. The combination of the two problems (i.e non-stationarity and partial observation) makes the problem non-solvable without using approximations.

2.3 Incremental gradient reinforcement learning

We propose to find approximate solutions by using incremental gradient RL. The main idea of this methodology is to progressively scale up with the complexity of the problem. Agents run their own local version of RL, here a gradient descent described in [Baxter and Bartlett, 1999].

Stochastic behaviors

In the context of partially observed MDP, stochastic behaviors perform better than deterministic policies (see [Singh *et al.*, 1994]). As previous experiments (see [Buffet *et al.*, 2001]) with Q-Learning were not conclusive to this regard, we chose to work with a gradient descent reinforcement learning algorithm especially suited for stochastic policies. This algorithm, designed by Baxter [Baxter and Bartlett, 1999] is detailed in Section 3.2.

Incremental learning

To speed up learning and reduce the problems of complexity and credit assignment, we propose a methodology for incremental learning. The most obvious way is to use :

- **growing number of agents** : learning starts with a small number of agents, each learning its own strategy. There must be enough agents to solve the problem. Then, more agents are added, with initial policies taken from the original agents and refined through learning if needed.

Incremental learning is also possible along the complexity dimension of the problem. Agents are pitted against harder and harder tasks. First tasks are "near" (in term of number of actions) positive reinforcement positions, then further and further away. So, we use :

- **progressive tasks** : learning begins with a very simple version of the task to be executed, or with the agent being heavily guided to solve the task. Then, as learning progresses, the task is made harder usually by giving more freedom of action to the agents

2.4 On local perception

We use situated agents with local perceptions to reduce the complexity of the problem as, very often, centralized problems are often too huge to be solved (see Section 3.1). This means that an agent does not know the global state of the world, which makes RL difficult to use.

However, local perceptions are a prerequisite for incremental learning, as behaviors of the agents can be based on the

"nearby" world. Thus, they scale immediately with the number of agents, the dimension of the world or the complexity of the task.

3 Experimenting with incremental learning

An application of this general notion of incremental learning is given in the experiments described here. After a short description of the problem, we give the details of the incremental tasks we used : harder tasks first and then more agents.

3.1 Problem description

The task chosen involves agents (either yellow or blue) in a grid world whose goal is to push yellow cubes against blue ones[1]. When two agents coordinate their movements to attain this goal -pushing *together* a pair of cubes- both cubes temporarily disappear. Simultaneously, agents responsible for this fusion receive a positive reward. The goal is to merge as many cubes as possible in a given time.

Considering our agents' abilities, they simply have four possible actions corresponding to moving North, East, South and West (they always try to move). Agents can push other agents and other cubes so the consequences of their actions are stochastic, depending on which constraints will be considered first.

Agents' perceptions, as shown on figure 3.1, are :

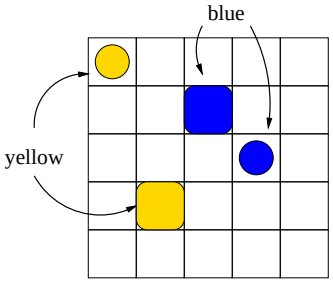
- `dir(oa)` : direction of nearest agent from opposite color `(N-E-S-W)`,
- `dir(cy)` : direction of nearest yellow cube `(N-NE-E-SE-S-SW-W-NW)`,
- `dir(cb)` : direction of nearest blue cube `(N-NE-E-SE-S-SW-W-NW)`,
- `near(cy)` : is there a yellow cube in one of the eight nearest positions `(true|false)`,
- `near(cb)` : is there a blue cube in one of the eight nearest positions `(true|false)`.

Combining these, there exists a maximum of 1024 observations (some combinations of perceptions are not possible). We reduce this number to 256 by using symmetries of the problem. This number is very small compared to the 15.249.024 states of the 8×8 totally observed centralized problem, and also *independent of the world's size*.

The reward function we have chosen eases the credit-assignment problem. Only the two agents identified as having originated the fusion of two cubes get a reward $(+5)$, whereas in other cases the reward is zero for all agents. A "global" reward would not be very coherent with agents having only local informations.

We conclude by some remarks about the simulation itself. To focus on the learning problem, we have implemented simple mechanisms to avoid some non-significant problems. For example, cubes cannot go on border cells of the grid. Similarly, when cubes reappear on the grid, they can only be put on inner cells.

[1]Colors of agents and cubes are not related in this problem.



agent	dir(cy)	dir(cb)	dir(oa)	near(cy)	near(cb)
yellow	S	E	SE	no	no
blue	W	NW	NW	no	yes

cy : yellow cube - cb : blue cube - oa : other agent

Figure 1: perceptions' examples (two agents in a simple world)

3.2 Learning algorithm used

As said earlier, each agent uses its own gradient descent algorithm to learn its policy . We precisely use an on-line version of the algorithm.

As proposed by Baxter [Baxter and Bartlett, 1999], a policy π depends on a set of parameters Θ. This leads to an expected utility $V(\pi_\Theta) = V(\Theta)$. The gradient descent permits [2] to find a locally optimal policy by finding Θ^* that maximizes $V(\Theta)$.

The set of parameters we chose is $\Theta = \{\theta(s,a), s \in \mathcal{S}, a \in \mathcal{A}\}$, with $\theta(s,a)$ a real valued parameter. The policy is defined by the probabilities of making action a in state s as :

$$\pi_\Theta(s,a) = \frac{e^{\theta(s,a)}}{\sum_{b \in \mathcal{A}} e^{\theta(s,b)}}$$

4 The 2-agent and 2-cube case

4.1 Reduced problem

The first step in our incremental learning scheme is to use small sized world with a minimal number of agents : one agent and one cube of each color. Then, beginning with positive reinforcement situations, the agents will face harder and harder problems.

4.2 Progressive task

For our problem, there is only one situation which leads to a positive reward. This situation is thus the starting point of incremental learning. Agents face a sequence of tasks to learn before ending in a standard 8×8 environment where they keep learning.

To be more precise, we define a *try* as a sequence of n steps[3] beginning in a given situation. This *try* must be repeated sufficiently (N times) to be useful. This succession of *tries* will be called an *experiment* for our agents. The trainer has to define a sequence of progressive *experiments* to help learning. Following [Asada *et al.*, 1996], we designed a training sequence where starting situations are more and

[2]It is true if V only depends on this agent's policy, which is not the case in a MAS.

[3]In a *step*, all agents make one move simultaneously.

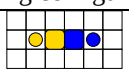
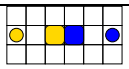
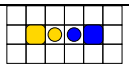
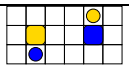
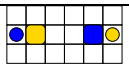
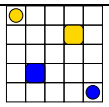
Starting configuration	n (moves)	N (tries)
	6	750
	6	500
	10	750
	20	750
	20	750
	100	75
	100	75

Table 1: The sequence of experiments we used for incremental learning.

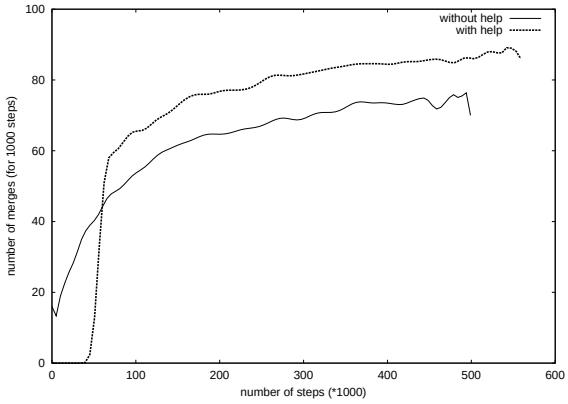


Figure 2: 2 agents, 2 cubes : incremental vs. raw learning

more "distant" from the goal, as Asada showed the efficiency of such a method.

Table 1 shows a sequence of experiments we used to help our agents in their training. The first *starting configuration*, on a 6×3 world, needs only one move to reach the goal, each agent pushing toward the cubes. However, they have up to 6 steps to reach it (so they can explore different moves), and they can try this 750 times.

Figure 2 shows the evolutions in the pair of agents' efficiency whether using or not progressive learning. The curve showing the efficiency of learning after a period of assisted training only begins after the 60000 steps of this training. Once this delay elapsed, notably better performances are reached with the assistance provided by our method.

cubes ↓	agents 2	4	8	16	20
2	**40.4**	30.0	20.0	12.7	11.0
4	7.6	17.1	**17.5**	13.9	12.9
6	3.4	11.2	14.7	15.7	**16.5**
8	1.9	8.6	13.5	15.9	**18.0**
10	1.6	6.7	11.0	17.7	**20.6**

Table 2: Average efficiencies
Number of merges for 1000 steps

5 More agents and more cubes

5.1 The next step in incremental learning

Until now, only two agents were used. More could be necessary to reach our goal in a world with more cubes.

In this section, we first have a look at the efficiency of a policy learned with $2a2c$ and used with more agents and more cubes, and then we let agents' policies evolve in this more populated worlds.

Note : Here, we use agents that have either already learned a policy through the *2a2c* case with help or have never learned anything at all.

5.2 Influence of the number of agents and cubes

In this part of our work, the first interest was to check the efficiency of different numbers of agents having to deal with different numbers of cubes (taking the same number of agents (or cubes) of each color). In this experiment, we use agents whose fixed behaviors have been learned in the $2a2c$ case, which is not supposed to be very efficient with more cubes. Nevertheless it gives them enough good reactions to have some good results.

Several tests were carried out with 2, 4, 6, 8 or 10 cubes and 2, 4, 8, 16 or 20 agents (always in a world of size 10×10). We used series of 1000 consecutive steps, these series beginning in random configurations. But as blocking situations could sometime occur, 100 such series were made in each case.

Table 2 gives the average efficiency in each of the twenty-five cases (the efficiency is the number of merges made in 1000 steps). It quickly shows that there seems to be an optimal number of agents for a given number of cubes as denoted by bold numbers in Table 2.

A growing number of agents improves the results, until they cramp each other and bring too many coordination problems. With a growing number of cubes, the results are improved only up to an optimal number beyond which agents hesitate on which cubes to work with and fall easily into oscillatory sequences of moves.

After another set of tries, in the same conditions but with policy-less agents, it appears that agents with a random behavior have always bad results, whatever their number.

5.3 Incremental learning along number of agents

With agents whose fixed behaviors have been learned in the 2 agents and 2 cubes case (*2a2c-policies*), we have seen that a growing number of cubes induces problems that can be solved

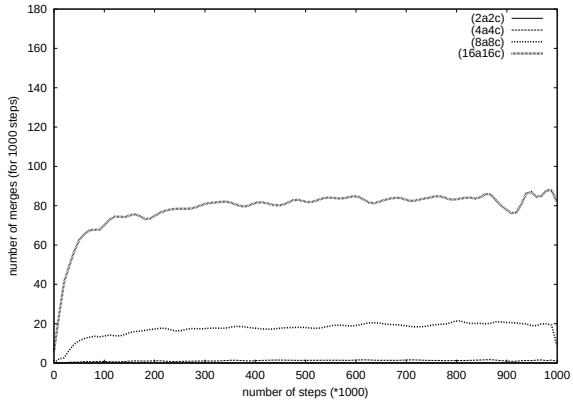


Figure 3: evolution while learning from scratch
(The $(4a4c)$ and $(2a2c)$ curves are practically confounded with abscissa axis.)

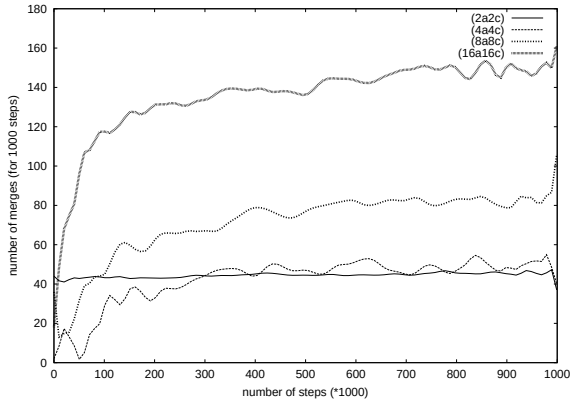


Figure 4: evolution while learning from *2a2c behaviors*

with a sufficient number of agents. The incremental learning solution is to improve the policies by keeping on learning with more agents.

Figures 3 and 4 show the efficiencies evolving with time in different learning cases : different number of objects, and agents learning from scratch or not. As shown on figure 3, agents learning from scratch have a slow evolution if there are only a few objects (agents and cubes) in the grid-world. But their learning speed rises with the rate $n_{objects}/size_{grid-world}$. When a group of agents seems to have reached a maximum efficiency, their results are better than the results obtained in Section 5.2 with the same situation but fixed policies. This confirms the interest of adaptation through learning.

One can notice that the optimal efficiency reached by agents using incremental learning is by far better than the performances reached using learning from scratch. As forecast, agents using *2a2c behaviors* come with some useful knowledge that allows them to find a better local optimum.

6 Discussion and similar works

6.1 On explicit coordination

Our experiments show that our work can benefit from addressing explicitly coordination problems. In particular, it is often the case that two agents, when placed in a world with more than two cubes, are not able to coordinate so as to push the "same pair of cubes". Moreover, when too many agents are present, they easily work at cross-purpose.

In [Boutilier, 1996], Boutilier has studied this problem from a theoretical point of view, using Multi-agent Markov Decision Processes. In his framework, each agent can compute a globally optimal policy where only coordination problems are to be solved, i.e. when the optimal joint action is not unique. Such coordination can be reached using social laws, communication or a learning mechanism. However, this centralized planning framework can not be used for decentralized RL. Other works, like [Hu and Wellman, 1998], more oriented towards reinforcement learning, settle on game-theory and Nash equilibria to solve coordination problems. Once again, these theoretical works are difficult to implement, especially with more than two agents.

Another field of research deals with explicitly taking into account other agents when learning, or planning, in a MAS (see [Carmel and Markovitch, 1996] or [Vidal and Durfee, 1997]). Usually, an agent builds a model of the other agents, so as to estimate their next actions. But, in the general case, it seems that not modeling the others is better than having a bad model and modelization leads to more complex agents.

Communication could also be used to solve the problem of explicit coordination. But attention must be paid to the fact that communication in itself has a cost [Xuan *et al.*, 2000]. Is it worthwhile to pay this cost whereas, simply putting more agents seems enough to increase the performances, even if the coordination problem of two particular agents is not really and explicitly solved ? This is one of the questions we would like to study in the near future. On the same subject, we would like to investigate the use of short term memory to provide our agent with attention mechanisms and help them to coordinate.

6.2 Reward

Matarić, in her work with multi-robots systems (see [Matarić, 1997]), took the option of adapting the reward function and the agents' actions and perceptions to use a simple reinforcement algorithm. The reward is not the kind of boolean process we have (no reward / big reward at goal) but rather a smoother reward adapted to give very frequent hints to each individual agent. By using behaviors, that can be seen as macro-actions, the size of state and action space is greatly reduced and learning is easier. On the other hand, this kind of specialization is very task-dependent whereas our agents could be made to learn another task without changing very much the agents or the learning process.

The COIN framework [Wolpert *et al.*, 1999] (COllective INtelligence) is similar to our framework : designing MAS through learning. The main ideas behind their work is to adapt reward functions and clusters of agents to build a *subworld-factored system*. Dealing with our problem , it

would mean creating only one sub-group sharing the same reward, and other experiments we conducted show that learning becomes difficult and slow. Whereas this framework is useful for finding "independent" sub-groups of agents, our ideas seem better adapted for in-group learning.

6.3 Automatic incremental learning

As explained in 2.3, we need to decompose the problem at hand in more and more complex tasks. The starting points of this decomposition are positive reward situations. The decomposition process could be automatically performed online by the learning agents. In addition to classical methods, the agents could randomly explore their environment, mark "interesting situations" and then work around them. Knowledge thus learned could then be transfered to other agents.

7 Conclusion

In this paper we have addressed the problem of automatically designing large Multi-Agent Systems. Our framework is based on each individual agent using a Reinforcement Learning algorithm to adapt its behavior to the desired global task. This learning problem is generally unsolvable because of its decentralized aspect and more classical limitations like partial observability of state, credit assignment and combinatorial explosion. To that effect, we have emphasized the use of an incremental learning scheme where more and more agents are faced with harder and harder problems. This method is very generic as it can be adapted to any task without adapting the agents to the particular problem at hand.

We have tested our approach on a simulated environment where agents have to coordinate in order to reach a shared goal : the fusion of different colored cubes. The experiments give credit to our framework as they show the efficiency of incremental learning. Incremental learning leads to better learning rates than raw unassisted learning. Furthermore, it is more efficient to learn a more complex task after an initial stage of incremental learning than learning directly this more complex task from scratch. As a whole, our framework led us to conclude that "the more the better".

Still, there is room for improvement. We have discussed several ways to overcome these limitations, like using communication for addressing explicit coordination, short-term memory and intentions to deal with oscillatory behaviors and policy-search reinforcement learning as a possibly more adequate learning algorithm. We have also considered modeling the behavior of the other agents as a mean to predict their actions and thus improve the behavior of agents.

8 Acknowledgments

We are particularly thankful to Bruno Scherrer, Iadine Chadès and Vincent Chevrier for numerous discussions and their invaluable help in writing this paper.

References

[Asada *et al.*, 1996] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, 23:279–303, 1996.

[Baxter and Bartlett, 1999] J. Baxter and P. Bartlett. Direct gradient-based reinforcement learning: I. gradient estimation algorithms. Technical report, The Australian National University, Canberra, Australia, 1999.

[Bernstein *et al.*, 2000] D.S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of markov decision processes. In *Proc. of the 16th Conf. on Uncertainty in Artificial Intelligence*, 2000.

[Boutilier, 1996] C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proc. TARK'96, De Zeeuwse Stromen, The Netherlands*, 1996.

[Buffet *et al.*, 2001] O. Buffet, A. Dutech, and F. Charpillet. Incremental reinforcement learning for designing multi-agent systems. In *Proc. of Agents'01*, 2001.

[Carmel and Markovitch, 1996] D. Carmel and S. Markovitch. Opponent modeling for learning in multi-agent systems. In *Proceedings of IJCAI'95 Workshop*. Springer, 1996.

[Dutech, 2000] A. Dutech. Solving pomdp using selected past-events. In *Proc. of the 14th European Conf. on Artificial Intelligence, ECAI2000*, 2000.

[Ferber, 1999] J. Ferber. *Multi-Agent Systems. An introduction to Distributed Artificial Intelligence.* John Wiley & Sons Inc., New York, 1999.

[Hu and Wellman, 1998] J. Hu and M. Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In *Proc. of ICML-98*, 1998.

[Littman *et al.*, 1995] M. Littman, A. Cassandra, and L. Kaelbling. Learning policies for partially observable environments: scaling up. In *Proc. of ICML-95*, 1995.

[Matarić, 1997] M. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.

[Singh *et al.*, 1994] S. Singh, T. Jaakkola, and M. Jordan. Learning without state estimation in partially observable markovian decision processes. In *Proc. of ICML-94*, 1994.

[Stone and Veloso, 2000] P. Stone and M. Veloso. Multiagent systems: a survey from a machine learning perspective. *Autonomous Robotics*, 8(3), 2000.

[Sutton and Barto, 1998] R. Sutton and G. Barto. *Reinforcement Learning*. Bradford Book, MIT Press, 1998.

[Vidal and Durfee, 1997] J. Vidal and E. Durfee. Agent learning about agents: a framework and analysis. In S. Sen, editor, *Collected papers from the AAAI-97 workshop on multiagent learning*. AAAI Press, 1997.

[Wolpert *et al.*, 1999] D. Wolpert, K. Wheeler, and K. Tumer. General principles of learning-based multi-agent systems. In *Proc. of Agents'99, Seattle*, pages 77–83, 1999.

[Xuan *et al.*, 2000] P. Xuan, V. Lesser, and S. Zilberstein. Communication in multi-agent markov decision processes. In *Proc. of ICMAS Workshop on Game Theoretic and Decision Theoretic Agents*, 2000.

Robot Weightlifting By Direct Policy Search

Michael T. Rosenstein and Andrew G. Barto
Department of Computer Science
University of Massachusetts
Amherst, MA 01003-4610
{mtr, barto}@cs.umass.edu

Abstract

This paper describes a method for structuring a robot motor learning task. By designing a suitably parameterized policy, we show that a simple search algorithm, along with biologically motivated constraints, offers an effective means for motor skill acquisition. The framework makes use of the robot counterparts to several elements found in human motor learning: imitation, equilibrium-point control, motor programs, and synergies. We demonstrate that through learning, coordinated behavior emerges from initial, crude knowledge about a difficult robot weightlifting task.

1 Introduction

Humans exhibit remarkable proficiency at complex motor tasks such as handwriting, juggling, and machine assembly. One way to characterize our success with such tasks is that we have a mastery of redundant degrees of freedom, cf. Bernstein [1967]. The ease with which humans achieve motor coordination contrasts sharply with that of robots, where "simple" motor tasks such as walking and throwing pose challenges to roboticists and artificial intelligence researchers. The typical solution of path planning and trajectory tracking works well for highly structured problems but not for situations with complex, nonlinear dynamics, with inadequate models, and with no detailed knowledge of the best solution. In this paper we examine a form of trial-and-error learning as a means for motor skill acquisition. Our results involve a simulated, three-link robotic arm that learns to raise heavy weights through efficient, timely use of its actuators.

Algorithms for trial-and-error learning typically fall near one of two extremes: those that take advantage of structure in the problem, and those that take advantage of structure in the solution. We have in mind the reinforcement learning problem where an agent interacts repeatedly with its environment and attempts to learn an optimal policy, i.e., a mapping from states to actions that maximizes some performance criterion. Algorithms that focus on the solution often make direct adjustments of the current policy, whereas methods that focus on the problem tend to build an intermediate representation, such as a value function, that forms the basis for policy improvement. In any case, the distinguishing features of trial-and-error learning are the exploratory activity and the simple evaluative feedback that reinforces successful behavior.

Reinforcement learning algorithms such as TD(λ) [Sutton, 1988] and Q-learning [Watkins and Dayan, 1992] are particularly well-suited to take advantage of structure in the problem. Such algorithms use a value function to capture regularities in the observed rewards and state transitions that implicitly define the task. However, reinforcement learning algorithms typically ignore structure in the solution by treating the policy as a featureless mapping from states to actions. Extensions to the basic reinforcement learning framework, such as function approximation, e.g., [Tsitsiklis and Van Roy, 1997], variable-resolution methods, e.g., [Moore and Atkeson, 1995], factored MDPs, e.g., [Boutilier *et al.*, 2000], and semi-Markov decison problems, e.g., [Sutton *et al.*, 1999], all constrain the form of the policy and, therefore, add strucutre to the solution. Nevertheless, with each of these extensions the focus remains on the value function rather than on the policy.

At the other extreme, methods for direct policy search, e.g., [Anderson, 2000; Baird and Moore, 1999; Baxter and Bartlett, 2000; Moriarty *et al.*, 1999], do away with intermediate representations of the policy and take advantage of constraints placed on the solution. For instance, evolutionary algorithms [Moriarty *et al.*, 1999] reinforce regularities in the parameterized policy (the "genetic material") by restricting "reproduction" to the fittest members of the population. And direct methods for function optimization [Swann, 1972] make the most of continuity by biasing their search toward promising regions of parameter space. However, such methods ignore structure in sequential decision problems by making policy adjustments on a trial-by-trial basis after long-term rewards are known, rather than on a per-action basis.

Both direct policy search and value-based methods often start with little or no prior structure and rely on extensive exploration to gather the information needed to build an optimal policy. Both approaches can benefit from domain knowledge. In this paper we utilize an algorithm for direct policy search because of its simplicity, not because we believe such methods are generally superior to value-based algorithms. Our goal is to show that biologically motivated structure can make trial-and-error learning an effective approach for a particular

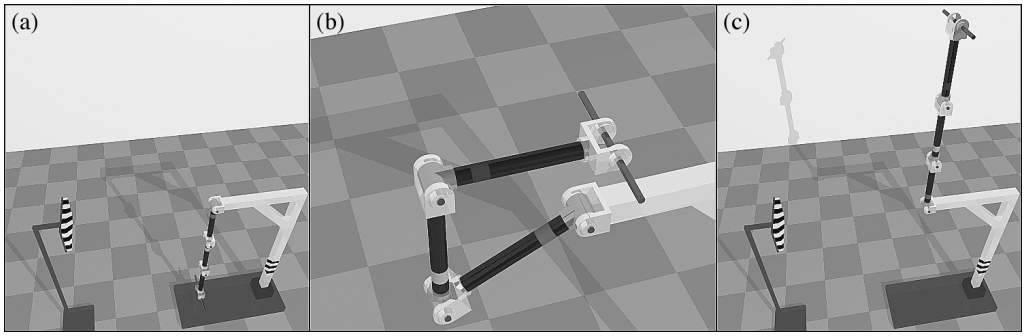


Figure 1: Simulated three-link robotic arm in several configurations with no payload: (a) start, (b) via point, and (c) goal. At the via point, the joint angles are -150, -120, and -90 degrees (proximal to distal).

class of control problems. Although we are certainly not the first to look to biology for inspiration with regard to machine learning and control, this work is novel in its combination of motor strategies for developing a coordinated robot.

2 Robot Weightlifting

By exploiting the dynamics of the task—gravity, inertia, elasticity, and so on—Olympic weightlifters solve a difficult coordination problem and raise more than twice their bodyweight over their heads. Our simulated robotic weightlifter, while loosely based on Olympic weightlifters, was inspired by the numerous acrobot-like examples studied by the machine learning and control engineering communities. Figure 1 shows the robot in several configurations. The task is to lift a payload from the straight-down, stable equilibrium to the straight-up, unstable equilibrium. Limited torque at each joint restricts the class of feasible solutions to those that execute a coordinated swing, i.e., those that exploit momentum and intersegmental dynamics. Kinematic redundancy, joint constraints, and obstacles (the striped object as well as the robot and its support structure) also complicate the task.

The robotic arm was simulated as a three-link frictionless pendulum with each link having length 1 m and and mass 1 kg. At each joint, torque was limited to $[-50,50]$ Nm, and at the middle and distal joints the range of motion was confined to $[-150,150]$ degrees. The equations of motion were generated iteratively using the Newton-Euler method [Walker and Orin, 1982] and solved numerically by Euler integration with a step size of 0.001 s. The robot was motionless at the start, and the goal was defined as a six-dimensional hypercube with velocity limits of ± 28.65 deg/s (± 0.5 rad/sec) and with position limits of ± 5 degrees centered on the goal configuration. At the end of each trial, i.e., when the robot's state entered the goal region, performance was quantified as the total integrated torque magnitude, and on those trials where the lifter exceeded its range of motion, contacted an obstacle, or failed to reach the goal within five seconds, the trial was terminated with the maximum cost of 500 Nm·s. In summary, the task was designed as a minimum-effort optimal control problem with both torque and kinematic constraints.

Optimal control theory provides a number of techniques for solving dynamic optimization problems such as our weightlifting task. For instance, Wang *et al.* [1999] used a gradient-based method starting from an initial feasible path to more than triple the recommended payload capacity of a Puma industrial robot. However, such methods usually require precise models (or system identification) and protracted offline computation before the robot attempts its first lift. In this paper we make no assumptions about the availability of a detailed model. Instead, we borrow several human motor learning strategies as a way to add the needed structure to an otherwise difficult machine learning problem.

3 Structured Policy Parameterization

Most research on human motor coordination has focused on control instead of learning, and so relatively little is known about the *mechanisms* of human motor learning. This leaves us, the robot designers, in a position to select from many available artificial intelligence and adaptive control techniques. Below, we describe a simple form of trial-and-error learning. However, our focus is not the choice of a specific algorithm but rather the motor control strategies that place constraints on that algorithm. In particular, we *design* a biologically motivated, parameterized policy that is easily seeded with prior knowledge and easily adapted through learning.

3.1 Biological Motivation

Our proposed solution begins with a sequence of *via points*, i.e., desired configurations for the robotic arm. (A similar approach was used by Morimoto and Doya [1998] for a robot stand-up task.) We regard each via point as the specification for a movement primitive that converges to the corresponding manipulator configuration. Such primitives are suggestive of an equilibrium-point style of motor control, e.g., [Feldman, 1986], whereby the endpoint of a movement is programmed and the spring-like properties of the muscles ensure convergence to that endpoint. The primary benefits of equilibrium-point control are that complex limb dynamics can be ignored and that higher motor centers need not be involved in the detailed activation patterns of numerous actuators. The drawback is that to explain complicated movements, equilibrium-point approaches give up their simplicity by requiring a *virtual trajectory*, or sequence of equilibrium points that induce the desired movement but are not targets for convergence.

For the weightlifting task, one possible implementation of equilibrium-point control involves the use of a single move-

ment primitive that brings the manipulator to the goal configuration. Indeed, with no payload and with no obstacle a simple linear feedback controller accomplishes the task, although this solution fails with payloads as small as 0.5 kg. Instead, we add a second movement primitive for the via point shown in Figure 1b. This via point represents the knowledge that certain configurations avoid the leftmost obstacle and also reduce the effective moment arm for the proximal joint. We assume the via point is obtained through imitation of a successful lift (cf. Schaal [1999]) or through instruction from a coach—a human programmer in our case. The via point is intended to convey crude path information, with no detailed knowledge of any successful trajectory.

Convergence first to the via point and then to the goal, extends the payload capacity to about 2 kg, beyond which adaptation is necessary. Thus, for the robotic weightlifter the advantage of equilibrium-point control is also its drawback: The robot's intrinsic dynamics are ignored for their difficulties as well as their benefits. Rather than turn to virtual trajectories as a way to exploit dynamics, we use a small number of movement primitives that act as the starting point for learning. The result is a hierarchical motor program that runs three feedback controllers: one for the via point and one for the goal, both adjustable, followed by another, fixed controller to regulate locally about the goal. This converts the closed-loop equilibrium-point solution to a "ballpark" open-loop solution [Greene, 1972; Schmidt, 1975] that handles minor errors at a lower, closed-loop level of control.

3.2 Implementation

To build the initial motor program, we first construct two proportional-derivative (PD) controllers, PD_1 and PD_2:

$$\text{PD}_i: \quad \boldsymbol{\tau}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \mathbf{W}_i[K_p(\boldsymbol{\theta}_i^* - \boldsymbol{\theta}) - K_v\dot{\boldsymbol{\theta}}], \qquad (1)$$

where $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$ are the joint positions and velocities, respectively, and $\boldsymbol{\tau} \in \Re^3$ is a vector of joint torques subject to saturation at the ± 50 Nm torque limit. In Eq. (1) $\mathbf{W}_i$ is a 3x3 gain matrix, $\boldsymbol{\theta}_i^*$ is the target equilibrium point, and $K_p = 1000$ Nm·rad^{-1} and $K_v = 250$ Nm·s·rad^{-1} are the nominal proportional and derivative gains, respectively. The target equilibrium points, $\boldsymbol{\theta}_1^*$ and $\boldsymbol{\theta}_2^*$, are initialized to the via point and goal configuration, respectively. Both $\mathbf{W}_1$ and $\mathbf{W}_2$ are initialized to the identity matrix, and so each PD controller initially acts as three uncoupled, scalar-output servomechanisms (one for each joint).

Next, the robot executes an "imitation" trial by running to convergence PD_1 followed by PD_2. (A threshold test on the position error establishes convergence.) At convergence we record t_1^* and t_2^*—the time elapsed for the corresponding controller since the start of the trial. Together t_1^* and t_2^* mark the switching times for the open-loop level of control. In particular, the program runs PD_1 from the start of each new trial (time $t = 0$) until $t = t_1^*$, then a switch is made to PD_2 which runs until $t = t_2^*$, followed by the third controller, PD_3, that runs until the trial terminates. PD_3 is a fixed copy of PD_2 that increases the flexibility of the learning system while providing convergence guarantees for movements close to the goal. However, PD_3 plays no role during the imitation trial.

The open-loop level of the motor program has two free parameters, the switching times, that we adjust by trial-and-error learning (described shortly). Together, PD_1 and PD_2 have 24 free parameters, six from $\boldsymbol{\theta}_i^*$ and 18 from $\mathbf{W}_i$, that we adapt with the same learning algorithm. As a form of shaping, we increase the payload at a nominal rate of 0.25 kg every 250 trials of learning. And with each new payload, learning proceeds in two phases: a shorter phase (50 trials) that adjusts the open-loop timing of the motor program, followed by a longer phase (200 trials) that adjusts the lower-level PD controllers.

3.3 Direct Policy Search

Table 1 summarizes the *simple random search* (SRS) algorithm developed for the robot weightlifting task. The algorithm performs random search in a K-dimensional parameter space, centered at a base point, $\mathbf{x}$. Perturbations, $\Delta\mathbf{x}$, are normally distributed with zero mean and standard deviation equal to the search size, σ. Each test point $(\mathbf{x} + \Delta\mathbf{x})$ is evaluated and the best observed point is kept as the algorithm's return value. For the weightlifting task, the evaluation function gives the total effort during a simulated trial with the new parameter settings. Updates to the base point are made by taking a step toward the most recent test point with probability β or toward the best observed point with probability $1 - \beta$. Thus, β provides an easy adjustment for the exploration-exploitation tradeoff, with the extremes of a pure random walk ($\beta = 1$) and of movement along a rough estimate of the gradient ($\beta = 0$). Even with β set to zero, considerable exploration is possible for large values of σ, which decays by a factor γ after each iteration, to the minimum σ_{min}.

input
  initial point $\mathbf{x} \in \Re^K$
  step size $\alpha \in$ [0,1]
  search strategy $\beta \in$ [0,1]
  search size $\sigma \geq 0$
  search decay factor $\gamma \in$ [0,1]
  minimum search size $\sigma_{min} \geq 0$
initialize
  $\mathbf{x}_{best} \leftarrow \mathbf{x}$
  $y_{best} \leftarrow \texttt{EVALUATE}(\mathbf{x}_{best})$
repeat
  1. $\Delta\mathbf{x} \leftarrow \mathbf{N}(\mathbf{0}, \sigma)$
  2. $y \leftarrow \texttt{EVALUATE}(\mathbf{x} + \Delta\mathbf{x})$
  3. **if** $y < y_{best}$
  4. $\quad \mathbf{x}_{best} \leftarrow \mathbf{x} + \Delta\mathbf{x}$
  5. $\quad y_{best} \leftarrow y$
  6. $\mathbf{x} \leftarrow \begin{cases} \mathbf{x} + \alpha \cdot \Delta\mathbf{x}, & \text{with prob. } \beta \\ \mathbf{x} + \alpha[\mathbf{x}_{best} - \mathbf{x}], & \text{with prob. } 1 - \beta \end{cases}$
  7. $\sigma \leftarrow \texttt{max}(\gamma\sigma, \sigma_{min})$
until convergence or number of iterations too large
return $\mathbf{x}_{best}$

Table 1: The simple random search (SRS) algorithm. For the motor program, $\mathbf{x} \in \Re^2$ at the open-loop level of control and $\mathbf{x} \in \Re^{24}$ at the closed-loop level.

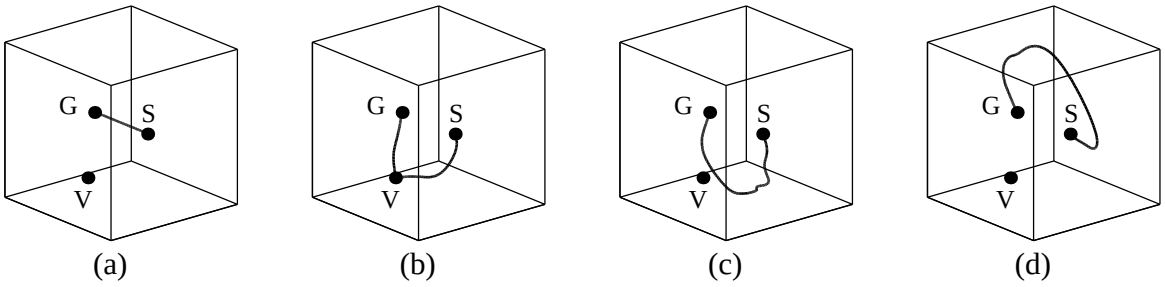


Figure 2: Configuration-space trajectories for (a) the simple equilibrium-point solution with no payload, no learning, and no obstacle, (b) the "imitation" trial with no payload, (c) a representative solution with a 4.5 kg payload, and (d) an unexpected solution with an 9.25 kg payload. *S*, *V*, and *G* denote the start, via-point, and goal configurations, respectively. The SRS algorithm step size and search strategy parameters were set to $\alpha = 0.3$ and $\beta = 0$, respectively. For the motor program, the initial search size was $\sigma = 0.10$ with a minimum of $\sigma_{min} = 0.01$ and a decay factor of $\gamma = 0.95$. For the PD controllers, the corresponding parameter values were $\sigma = 0.05$, $\sigma_{min} = 0.01$, and $\gamma = 0.99$.

The SRS algorithm has several properties that make it a nice choice for trial-and-error learning. First, SRS is easy to implement for a wide variety of optimization problems. Like other direct search methods [Swann, 1972], the SRS algorithm needs no derivative information—an important feature when the cost function is non-differentiable or when the gradient is difficult to estimate (as with deterministic policies subject to noise). The algorithm also has some neurobiological support, albeit speculative [Anderson, 1998]. And compared to "pattern search" algorithms [Beveridge and Schechter, 1970], such as the simplex method, SRS makes rapid progress in high-dimensional spaces where some algorithms first require numerous exploratory moves to establish a search direction or to build a simplex, for instance.

4 Learning To Exploit Dynamics

With no payload, the robotic weightlifter can use a number of qualitatively different solutions to reach the goal configuration. For example, the simple equilibrium-point solution *with no obstacle* (Figure 2a) follows a direct path from the start to the goal, whereas the "imitation" trial (Figure 2b) takes a longer path through configuration space, first converging to the via point. As the payload increases, the space of feasible solutions shrinks, and so the via point represents an attempt to start the learning system at a favorable place. The "standard" solution after learning (Figure 2c) passes by the via point while coordinating the joints to exploit the robot's intrinsic dynamics. (The standard solution is robust to sensor noise as well as variability in the via point.) Figure 2 also shows an unexpected "reversal" solution, where the robot moves through an entirely different region of configuration space (details below).

As indicated in Section 3.2, each PD controller initially behaves as three independent, linear servomechanisms—one for each joint—but the learning algorithm transforms them into the robot analogue of a synergy [Bernstein, 1967; Greene, 1982] or "smart" nonlinear controller that accounts for intersegmental effects. (Although Eq. (1) describes a linear controller, saturation of the output represents a set of nonlinear constraints, possibly inactive, that the learning system can exploit.) To examine the coupling of the individual controllers, we quantify joint coordination as

$$C(\mathrm{PD}_1, \mathrm{PD}_2) = 2 - [D(\mathbf{W}_1) + D(\mathbf{W}_2)], \qquad (2)$$

where D is the following measure of diagonal dominance of a PD controller gain matrix:

$$D(\mathbf{W}) = \frac{\sum_j |w_{jj}|}{\sum_{j,k} |w_{jk}|} \qquad (3)$$

Figure 3 shows the change in C throughout learning—starting with no coupling (both $\mathbf{W}_1$ and $\mathbf{W}_2$ diagonal) for the initial trial and increasing to about 1.0 where half the "mass" in the gain matrices falls along the main diagonals. With no payload, the increase in coupling mirrors the drop in effort, and with all but the lightest payloads, we observe a statistically significant increase in coupling over the no-payload condition. Thus, active coordination by the control system appears to play an important role for the weightlifting task, although the functional significance of these results remains a question for future work.

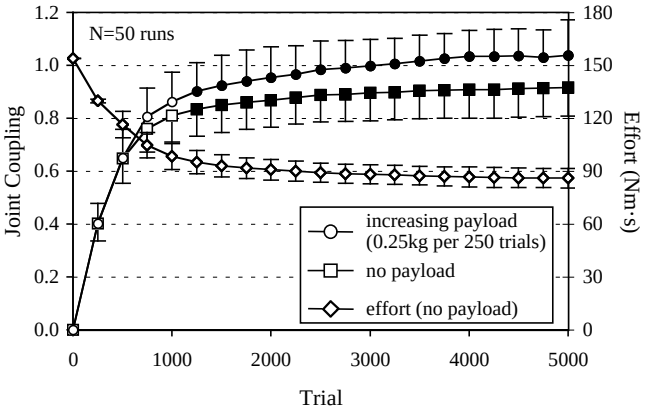


Figure 3: Effects of payload and learning on controller joint coordination (averaged over 50 runs). Solid markers denote statistically significant differences ($p<0.01$) at the corresponding trial number.

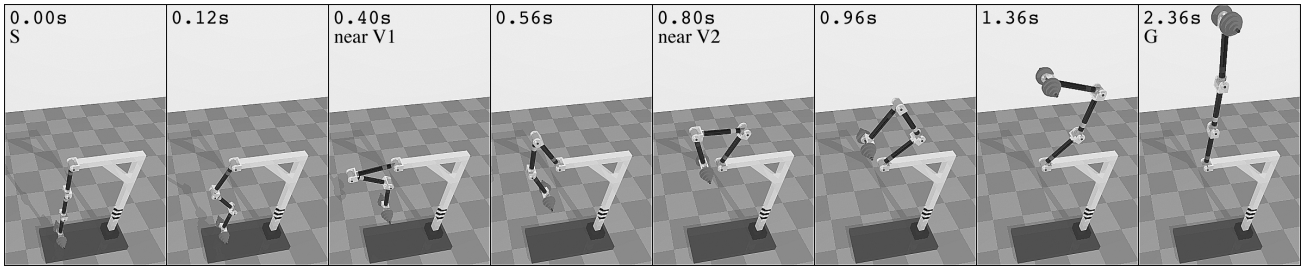


Figure 4: The reversal solution with a 9.25 kg payload. The manipulator was nearest the two via points, *V1* and *V2*, at t=0.40 s and t=0.80 s.

Early in learning, initial motion at the middle and distal joints induces counterclockwise movement at the proximal joint—despite the action of the PD controller which attempts to drive the upper arm clockwise, away from the support structure and directly toward the via point. In other words, passive mechanincal coupling conflicts with the initially uncoupled, active control. Through subsequent learning, however, the best solutions exploit intersegmental dynamics and initial swing (i.e., momentum) by reversing the sign of the "shoulder" torque for the first 100 to 200 ms.

On rare occasions—less than one percent of all runs—the learning system discovered the "reversal" solution shown first in Figure 2d and again, with more detail, in Figure 4. Throughout much of the movement, the positions of the middle and distal joints have opposite sign with respect to the standard solution. In effect, the reversal solution abandons the prior knowledge supplied by the via point. Interestingly, we observed the reversal solution only for those experiments with increased variability, involving sensor noise and a large initial search size, σ.

Once we know of an alternative solution, we can use the same approach to policy parameterization to offer new "coaching" to our robotic weightlifter. For instance, we can encourage initial counterclockwise swing by inserting an extra via point (and an extra PD controller) prior to the one shown in Figure 1b. Separately, we can also encourage the reversal solution by designing two new via points as depicted in Figure 4 (frames three and five). These alternatives resulted in statistically significant improvements (p<0.001) as summarized in Figure 5. Thus, by injecting simple, approximate knowledge, we turned the reversal solution from a rare occurrence into a new standard by which to gauge future performance.

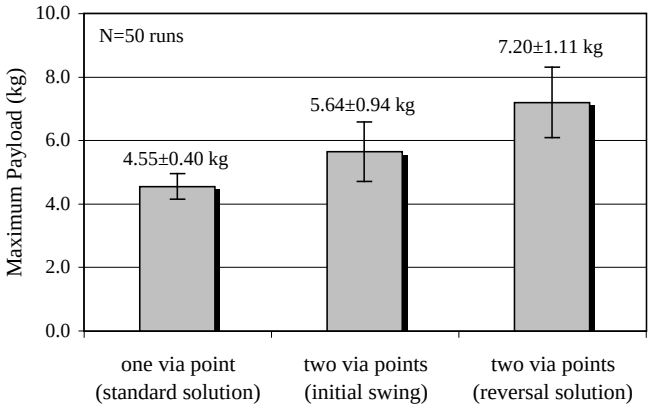


Figure 5: Effects of "coaching" on maximum payload lifted successfully after learning (averaged over 50 runs).

5 Conclusions

We attribute much of the success of the SRS algorithm to the prior structure supplied by both the parameterized policy and the via points. We suspect that more sophisticated forms of direct function optimization, e.g., [Swann, 1972], will yield improvements in terms of learning time, but that the qualitative results will remain the same. The more interesting possibility for future work is the use of gradient-based methods for continuing control tasks—whether a direct policy search algorithm, e.g., [Baxter and Bartlett, 2000], or a value-based approach, e.g., [Sutton *et al.*, 2000]. Such methods are particularly important for extending the basic approach outlined in this paper to non-episodic tasks with either occasional or continuing rewards.

For robotic manipulators, the classic solution to a movement problem involves, not reinforcement learning, but rather the construction of a kinematic trajectory that avoids obstacles and singularities. The underlying assumption is that the robot's intrinsic dynamics are something to compensate for, instead of something to exploit. Even for optimized trajectories, imprecise dynamic models often lead to overly conservative acceleration constraints and, therefore, to sub-optimal movements [Craig, 1988]. Rather than use adaptive control techniques to deal with inaccurate models, in this paper we gave up the stability guarantees associated with control engineering methods and developed a learning framework motivated by human motor control. As part of future work, we plan to test this framework on a real, non-planar robot with seven degrees of freedom. Like other reinforcement learning methods, our approach is geared toward optimization; thus, the resulting policies are inherently ones that exploit dynamics.

As mentioned in Section 3.1, one criticism of equilibrium-point styles of motor control is the need for virtual trajectories to explain complicated multi-joint movements. Van Ingen Schenau *et al.* [1995] also argue that equilibrium-point models are kinematic in nature and, therefore, ill-suited for tasks that exploit dynamics or require control of contact forces. Although we agree with these criticisms, equilibrium-point control nevertheless plays a key role in the present

work. In particular, the movement primitives (i.e., the via points and the goal configuration) supply the hierarchical motor program with an initial kinematic solution that the learning algorithm then transforms into one that exploits, rather than cancels the dynamics.

Acknowledgments

We thank Theodore Perkins and Richard van Emmerik for comments and helpful discussions. This research was supported by the National Science Foundation under Grant No. IRI-9720345.

References

[Anderson, 1998] Russell W. Anderson. Biased random-walk learning: a neurobiological correlate to trial-and-error. In Omid Omidvar and Judith Dayhoff, editors, *Neural Networks and Pattern Recognition*, pages 221–244. Academic Press, San Diego, CA, 1998.

[Anderson, 2000] C. W. Anderson. Approximating a policy can be easier than approximating a value function. Technical Report CS-00-101, Colorado State University, 2000.

[Baird and Moore, 1999] L. Baird and A. Moore. Gradient descent for general reinforcement learning. In Michael S. Kearns, Sara A. Solla, and David A. Cohn, editors, *Advances In Neural Information Processing Systems 11*, pages 968–974. The MIT Press, Cambridge, MA, 1999.

[Baxter and Bartlett, 2000] Jonathan Baxter and Peter L. Bartlett. Reinforcement learning in POMDPs via direct gradient ascent. In *Proceedings of the Seventeenth International Conference on Machine Learning,*, 2000.

[Bernstein, 1967] N. A. Bernstein. *The Co-ordination and Regulation of Movements*. Pergamon Press, Oxford, 1967.

[Beveridge and Schechter, 1970] Gordon S. G. Beveridge and Robert S. Schechter. *Optimization: Theory and Practice*. McGraw-Hill Book Co., New York, 1970.

[Boutilier *et al.*, 2000] Craig Boutilier, Richard Dearden, and Moises Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49–107, 2000.

[Craig, 1988] John J. Craig. *Adaptive Control of Mechanical Manipulators*. Addison-Wesley Publishing Company, Reading, MA, 1988.

[Feldman, 1986] A. G. Feldman. Once more on the equilibrium-point hypothesis (λ model) for motor control. *Journal of Motor Behavior*, 18:17–54, 1986.

[Greene, 1972] P. H. Greene. Problems of organization of motor systems. In R. Rosen and F. M. Snell, editors, *Progress In Theoretical Biology*, volume 2, pages 303–338. Academic Press, New York, 1972.

[Greene, 1982] P. H. Greene. Why is it easy to control your arms? *Journal of Motor Behavior*, 14(4):260–286, 1982.

[Moore and Atkeson, 1995] A. W. Moore and C. G. Atkeson. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning*, 21(3):199–233, 1995.

[Moriarty *et al.*, 1999] David E. Moriarty, Alan C. Schultz, and John J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276, 1999.

[Morimoto and Doya, 1998] J. Morimoto and K. Doya. Reinforcement learning of dynamic sequence: learning to stand up. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1721–1726, 1998.

[Schaal, 1999] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Science*, 3:233–242, 1999.

[Schmidt, 1975] Richard A. Schmidt. A schema theory of discrete motor skill learning. *Psychological Review*, 82(4):225–260, 1975.

[Sutton *et al.*, 1999] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

[Sutton *et al.*, 2000] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Muller, editors, *Advances In Neural Information Processing Systems 12*, pages 1057–1063. The MIT Press, Cambridge, MA, 2000.

[Sutton, 1988] Richard S. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.

[Swann, 1972] W. H. Swann. Direct search methods. In W. Murray, editor, *Numerical Methods For Unconstrained Optimization*, pages 13–28. Academic Press, New York, 1972.

[Tsitsiklis and Van Roy, 1997] J. N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning. *IEEE Transactions on Automatic Control*, 42:674–690, 1997.

[van Ingen Schenau *et al.*, 1995] G. J. van Ingen Schenau, A. J. van Soest, F. J. M. Gabreels, and M. W. I. M. Horstink. The control of multi-joint movements relies on detailed internal representations. *Human Movement Science*, 14(4–5):511–538, 1995.

[Walker and Orin, 1982] Michael W. Walker and David E. Orin. Efficient dynamic computer simulation of robotic mechanisms. *Journal of Dynamic Systems, Measurement and Control*, 104:205–211, 1982.

[Wang *et al.*, 1999] C.-Y. E. Wang, W. K. Timoszyk, and J. E. Bobrow. Weightlifting motion planning for a puma 762 robot. In *Proceedings of the IEEE 1999 International Conference on Robotics and Automation*, volume 1, pages 480–485, 1999.

[Watkins and Dayan, 1992] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3/4):279–292, 1992.

MACHINE LEARNING AND DATA MINING

INDUCTIVE LOGIC PROGRAMMING

OI-implication: Soundness and Refutation Completeness

Floriana Esposito, Nicola Fanizzi, Stefano Ferilli, Giovanni Semeraro
Dipartimento di Informatica, Università di Bari, Via Orabona, 4 I-70125 Bari, Italy
{esposito,fanizzi,ferilli,semeraro}@di.uniba.it

Abstract

Weakening implication by assuming the *object identity* bias allows for both a model-theoretical and a proof-theoretical definition of a novel and more manageable ordering relationship over clausal spaces. In this paper, we give two important results, namely the soundness and the refutation completeness (through a subsumption theorem) of the underlying derivation procedure, that make this relationship particularly appealing for inducing a generalization model for clausal search spaces.

1 Introduction

Logical implication, as a relationship on clausal spaces, is particularly hard to handle due to many negative results descending from its non-decidability [Schmidt-Schauss, 1988] and intrinsic complexity [Gottlob and Fermüller, 1993]. This is the reason why subsumption, being more tractable and efficiently implementable than stronger relationships (i.e., relationships that entail it), is the generalization model commonly exploited in *Inductive Logic Programming* (ILP).

Attempts have been made to weaken subsumption and implication in order to obtain more manageable ordering relationships between clauses. Consider, for instance, forms of *non-decreasing subsumption* such as the *weak subsumption* [Badea and Stanciu, 1999], or weaker but decidable forms of implication such as *T-implication* [Idestam-Almquist, 1995]. Indeed, the algebraic structure of the search space, and hence the refinement operators that can be defined, are affected by the choice of the underlying ordering relationship.

Weakening implication by assuming the *object identity* bias allowed for the definition of *OI-implication*, a novel relationship between clauses which makes the search space more manageable [Esposito *et al.*, 2000]. Specifically, it has been possible to prove the existence of *ideal* refinement operators for the search spaces ordered by the resulting form of logical implication, while this is not possible in standard clausal spaces [Nienhuys-Cheng and de Wolf, 1997]. This is particularly important for the efficiency of the refinement in search spaces with *dense solutions* [Badea and Stanciu, 1999]. Indeed, it has been shown as the computational complexity of the resulting operators is much more appealing than the one of the refinement operators defined for other search spaces.

Furthermore, the object identity framework for machine learning has been also exploited in the definition of operators performing inductive learning as well as abstraction and abduction, so to have a unique knowledge representation and reasoning framework for *multistrategy learning*.

In this paper, a proof-theoretic definition is adapted from [Esposito *et al.*, 2000] and a new model-theoretic definition of the implication under object identity is introduced. These definitions are shown to be equivalent. Indeed, two main results are demonstrated, namely the soundness of the derivation procedure compliant with the assumption of the mentioned bias and a subsumption theorem which allows for the demonstration that this procedure is also refutation-complete.

These results open new directions for the investigation of other interesting properties of this form of implication, such as the proof of a compactness theorem, the decidability and all the other algebraic properties of the search space, ordered by the relationship induced by this form of implication.

This paper is organized as follows. In Section 2, a semantics is defined, complying with the object identity assumption; hence, the definition of OI-implication is presented together with the proof of the soundness of the related proof procedure. Refutation completeness through the Subsumption Theorem, are proven in Section 3. Section 4 draws the conclusions of this research, outlining the ongoing developments.

2 OI-implication

In our framework, we adopt a representation language $\mathcal{L}$ expressing theories as *logic programs* made up of *clauses*. The basic notions about clausal representation can be found in [Lloyd, 1987; Nienhuys-Cheng and de Wolf, 1997].

The framework we propose relies essentially on the following assumption, which is commonly employed. For instance, defining the concept of bicycle, one would mention the two wheels among its components. Then, in order to rule out an instance of a different concept such as a monocycle, the identification of the two wheels should be avoided (a bicycle is two-wheeled by definition).

A thorough treatment of these issues for the Datalog clausal language is offered in [Semeraro *et al.*, 1998]. We want to extend this to a more expressive clausal language, without the restriction to function-free terms. In our setting we want to avoid the identification of terms.

Assumption 2.1 (Object Identity) *In a clause, terms denoted with different symbols must be distinct, i.e. they represent different entities of the domain.*

This notion can be viewed as an extension of Reiter's *unique names* assumption [Reiter, 1980]. Object identity is the basis for the definition of both an equational theory for the language clauses and a quasi-order upon them.

This assumption brings to a modified equational theory consisting in the addition of one axiom to an Equality Theory [Lloyd, 1987], namely:

$$\forall \neg(t_1 = t_2)$$

for all pairs (t_1, t_2) of distinct terms occurring in a clause.

Nienhuys-Cheng and de Wolf have embedded an equality theory in the semantics, for demonstrating the completeness of the SLDNF resolution, when dealing with the completion of normal programs, yielding the so called *E-Herbrand interpretations* [Nienhuys-Cheng and de Wolf, 1997]. Badea and Stanciu, in the definition of the weak subsumption [1999], deal with substitutions which do not identify literals. In this paper, we adopt a different approach, founded on a semantic basis. A similar approach to capturing the object identity assumption can be considered that of the *unique substitution semantics* [Khardon, 1999].

2.1 Semantics and Model-Theoretic Definition

Some definitions follow, specifying how the object identity assumption changes the semantics for the clausal representation. We start off by defining the relation between the terms of the language and the domain we deal with.

A *pre-interpretation J under object identity* of a first-order language is exactly a standard pre-interpretation:

- A non-empty set $\mathcal{D}$ called the *domain* of the pre-interpretation;
- Each constant in $\mathcal{L}$ is assigned to an element of $\mathcal{D}$;
- Each n-ary function symbol f is assigned a mapping J_f from $\mathcal{D}^n$ to $\mathcal{D}$.

Then, we need to map variables (and terms) to the objects of the domain:

Definition 2.1 *Let J be a pre-interpretation of $\mathcal{L}$ with domain $\mathcal{D}$. A* variable assignment under object identity *V w.r.t. $\mathcal{L}$ is a one-to-one mapping from the variables of $\mathcal{L}$ to the domain $\mathcal{D}$ of J. We use $V(X/d)$ to denote that the variable X is mapped to $d \in \mathcal{D}$ in the variable assignment V, and the other variables are mapped according to V. The* term assignment under object identity *Z in a formula ϕ w.r.t. J and V of terms in $\mathcal{L}$ is the following mapping from the terms in $\mathcal{L}$ to $\mathcal{D}$:*

1. *J maps each constant to an element of $\mathcal{D}$;*
2. *V maps each variable to a distinct element of $\mathcal{D}$;*
3. *If $d_1, \ldots, d_n$ are the elements of $\mathcal{D}$ to which the terms $t_1, \ldots, t_n$ are mapped, then the term $f(t_1, \ldots, t_n)$ is mapped to $J_f(d_1, \ldots, d_n)$, where J_f is the function from $\mathcal{D}^n$ to $\mathcal{D}$ assigned to the function symbol f by J, s.t. $\forall t_1, t_2 \in terms(\phi) : Z(t_1/d_1), Z(t_2/d_2)$ implies $d_1 \neq d_2$.*

By combining pre-interpretations and variable (term) assignments, we can specify the notion of interpretation under object identity:

Definition 2.2 *Given J a pre-interpretation of $\mathcal{L}$ with domain $\mathcal{D}$, an* interpretation I under object identity (OI-interpretation) *based on J assigns a mapping I_P from $\mathcal{D}^n$ to $\{T, F\}$ to each predicate symbol P.*
Given a variable assignment under object identity V, be Z the corresponding term assignment w.r.t. J and V. Then a formula ϕ in $\mathcal{L}$ has a truth value under *I defined as follows:*

1. *For each atom $A = P(t_1, \ldots, t_n)$, if $\forall i = 1, \ldots, n$: $d_i \in \mathcal{D}$ is assigned to t_i by Z, then the truth value of A under I and V is $I_P(d_1, \ldots, d_n)$;*
2. *The truth values of the formula ϕ obtained by using connectives is determined using the standard truth tables.*
3. *If $\phi = \exists X\psi$ then ϕ has truth value T under I and V if there exists $d \in \mathcal{D}$ for which ψ has truth value T under I and $V(X/d)$, where $V(X/d)$ is a legal variable assignment under object identity. Otherwise, ϕ has truth value F under I and V.*
4. *If $\phi = \forall X\psi$ then ϕ has truth value T under I and V if for all elements $d \in \mathcal{D}$ ψ has truth value T under I and $V(X/d)$, where $V(X/d)$ is a legal variable assignment under object identity. Otherwise, ϕ has truth value F under I and V.*

The standard notions of *tautology, contradiction, satisfiability* and *consistency* can be straightforwardly extended to the new semantics.

In ILP we focus our interest on clauses. Since they are closed formulas, the truth values do not depend on variable assignments. We can represent an interpretation as a set of ground literals, i.e. those that are true according with the interpretation.

Definition 2.3 *Given a closed formula ϕ, an OI-interpretation I is called a* model under object identity (OI-model) *for ϕ iff I satisfies ϕ. Given a set of closed formulas Σ, I is an OI-model for Σ iff it is an OI-model for all formulas $\phi \in \Sigma$.*

OI-interpretations are always standard interpretations. Conversely, we show with an example a case when an OI-model for a formula is not a model.

Example 2.1 *Let $C = p(X) \vee p(a)$. An OI-model for C, assumed the domain is $\mathcal{D} = \{a, b\}$, is represented by the Herbrand OI-interpretation $I = \{p(b)\}$.*
I is not a model for C in the standard meaning. Indeed, since $p(a)$ is stated as false, the truth of the clause depends on $p(X)$, where variable X is universally quantified. Under object identity, it is sufficient that $p(b)$ be true for $\forall X p(X)$ to hold. The same does not apply in the standard semantics where both $p(a)$ and $p(b)$ should be true.

We are now ready to define the form of implication that is compliant with the object identity assumption:

Definition 2.4 *Given a set of formulas Σ and a formula ϕ, Σ* OI-implies *ϕ or ϕ is a* logical consequence under object identity *of Σ (denoted with $\Sigma \models_{\text{OI}} \phi$) iff all OI-models I of Σ are OI-also models of ϕ.*

Thus, useful results hold also in this semantics:

Theorem 2.1 (Deduction Theorem) *Let Σ be a set of formulas and ϕ and ψ be formulas. Then $\Sigma \cup \{\phi\} \models_{\text{OI}} \psi$ iff $\Sigma \models_{\text{OI}} (\phi \rightarrow \psi)$.*

Proof. $\Sigma \cup \{\phi\} \models_{\text{OI}} \psi$ *iff*
all OI-models of $\Sigma \cup \{\phi\}$ *are OI-models of* ψ *iff*
all OI-models of Σ *are OI-models of* $\neg\phi$ *or of* ψ *iff*
all OI-models of Σ *are OI-models of* $\neg\phi \vee \psi$ *iff*
$\Sigma \models_{\text{OI}} (\phi \rightarrow \psi)$.

As a consequence, it is possible to prove also this result:

Theorem 2.2 *Let Σ be a set of formulas and ϕ be a formula. Then $\Sigma \models_{\text{OI}} \phi$ iff $\Sigma \cup \{\neg\phi\}$ is unsatisfiable.*

Proof. $\Sigma \models_{\text{OI}} \phi$ *iff (by Theorem 2.1)* $\Sigma \cup \{\neg\phi\} \models_{\text{OI}} \Box$ *iff*
all OI-models of $\Sigma \cup \{\neg\phi\}$ *are OI-models of* $\Box$ *iff*
$\Sigma \cup \{\neg\phi\}$ *is unsatisfiable under object identity.*

As concerns clauses, in the standard semantics it is possible concentrate the interest on Herbrand models only [Lloyd, 1987], This can be extended to our setting.

Proposition 2.3 *Let Σ be a set of clauses in a first order language $\mathcal{L}$. Then Σ has an OI-model iff Σ has an Herbrand OI-model.*

Proof.

($\Rightarrow$) *Suppose Σ has an OI-model I. Then we define the Herbrand OI-interpretation $\overline{I}$ with:*

1. *the same pre-interpretation of I and having as domain the Herbrand universe $\mathcal{U}_{\mathcal{L}}$;*
2. *Let P an n-ary predicate symbol occurring in Σ. Then we define the function $\overline{I}_P$ from $\mathcal{U}_{\mathcal{L}}$ to $\{F, T\}$ as follows: $\overline{I}_P(t_1, \ldots, t_n) = T$ if $P(t_1, \ldots, t_n)$ is true under I and $\overline{I}_P(t_1, \ldots, t_n) = F$ otherwise.*

($\Leftarrow$) *Obvious.*

2.2 Proof-Theoretic Definition and Soundness

Given a notion of subsumption under object identity, and using a subsumption theorem, we define implication under object identity proof-theoretically. The goal here is to define this relationship in a constructive way.

First, we have to define a form of resolution coping with the object identity assumption. To this purpose, we discuss some further properties required to substitutions in order to fulfill this assumption. In fact, a substitution can be regarded as a function mapping variables to terms. Hence, we require these functions to satisfy certain additional properties in order to preserve object identity.

Definition 2.5 *Given a set of terms T, a substitution σ is an* OI-substitution w.r.t. *T iff $\forall t_1, t_2 \in T : t_1 \neq t_2$ implies $t_1\sigma \neq t_2\sigma$.*

In the following, we will omit the set of terms T when it is obvious. Based on OI-substitutions, it is possible to define other related notions such as the *composition* of OI-substitutions, *ground* and *renaming* OI-substitutions, as well as *instance* clauses, *ground* clauses and *variants*.

In order to cope with the object identity assumption, a new relationship has been derived [Semeraro *et al.*, 1998] from the classic θ-subsumption. Its definition can be given by exploiting the notion of OI-substitution [Esposito *et al.*, 2000].

Definition 2.6 *Let C and D be two clauses. Then, C θ_{OI}-subsumes D iff there exists σ, an OI-substitution w.r.t. $terms(C)$, such that $C\sigma \subseteq D$.*

The notion of θ_{OI}-subsumption induces a quasi-order upon the space of clauses [Semeraro *et al.*, 1998]. Besides, a notion of equivalence for θ_{OI}-subsumption can be defined, that partitions the space of clauses in equivalence classes. Since θ_{OI}-subsumption maps each literal of the subsuming clause onto a single literal in the subsumed one, equivalent clauses under θ_{OI}-subsumption must have the same number of literals. Thus, a search space ordered by θ_{OI}-subsumption is made up of non-redundant clauses. This yields smaller equivalence classes than those in a space ordered by θ-subsumption.

Now, as we have done for the substitutions, we can specify a notion of a unifier fulfilling Assumption 2.1:

Definition 2.7 *Given a finite set of simple expressions S, we say that θ is an* OI-unifier *iff $\exists E\ \forall E_i \in S :\ E_i\theta = E$ and θ is an OI-substitution w.r.t. $terms(E_i)$. An OI-unifier θ for S is called a* most general OI-unifier (*mgu*$_{\text{OI}}$) *for S iff, for each OI-unifier σ of S, there exists an OI-substitution τ such that $\sigma = \theta\tau$.*

In this setting, clauses cannot have proper subsets as *factors*. Hence, this notion can be used only for standardizing apart the variables of the parent clauses before a step of resolution [Esposito *et al.*, 2000].

Definition 2.8 *Given the clauses C and D, suppose they are standardized apart. A clause R is an* OI-resolvent *of C and D iff, there exist $M \subseteq C$ and $N \subseteq D$ such that $\{M, \overline{N}\}$ is unifiable through the mgu*$_{\text{OI}}$ *θ and*

$$R = ((C \setminus M) \cup (D \setminus N))\theta$$

We write $R = \mathcal{R}_{\text{OI}}(\{C, D\})$.

This definition of OI-resolution follows Robinson's original definition [Robinson, 1965]. However, a different definition has been given, based on the one given in [Chang and Lee, 1973], taking into account one complementary pair of literals for each resolution step [Esposito *et al.*, 2000].

Definition 2.9 *Let T be a set of clauses. The* OI-resolution closure *of T, denoted by $\mathcal{R}^*_{\text{OI}}(\Sigma)$, is defined inductively:*
$(n = 0)\ \mathcal{R}^0_{\text{OI}}(\Sigma) = \Sigma$
$(n > 0)\ \mathcal{R}^n_{\text{OI}}(\Sigma) = \mathcal{R}^{n-1}_{\text{OI}}(\Sigma) \cup \{R = \mathcal{R}_{\text{OI}}(\{C, D\}) |\ C, D \in \mathcal{R}^{n-1}_{\text{OI}}(\Sigma)\}$
$\mathcal{R}^*_{\text{OI}}(\Sigma) = \mathcal{R}^0_{\text{OI}}(\Sigma) \cup \mathcal{R}^1_{\text{OI}}(\Sigma) \cup \cdots \cup \mathcal{R}^n_{\text{OI}}(\Sigma) \cup \cdots$
If there exists an OI-derivation *of C from Σ, this will be denoted with $\Sigma \vdash_{\text{OI}} C$ which is equivalent to $C \in \mathcal{R}^*_{\text{OI}}(\Sigma)$.*

It is interesting to consider also the case of *linear resolution* [Chang and Lee, 1973], which was originally exploited for the proof-theoretic definition of OI-implication [Esposito *et al.*, 2000]. If C can be derived by means of zero or more linear OI-resolution steps from the set of clauses Σ, we denote

this with $C \in \mathcal{L}^n_{\text{OI}}(\Sigma),\ n > 0$, where $\mathcal{L}_{\text{OI}}$ denotes the linear OI-resolution operator and $\mathcal{L}^*_{\text{OI}}$ stands for its closure.

In order to prove the *soundness* of the proof-procedure defined above, some preliminary results are needed.

Lemma 2.4 *Let C_1 and C_2 be clauses. If there exists R that is the OI-resolvent of C_1 and C_2 then it holds $\{C_1, C_2\} \models_{\text{OI}} R$.*

Proof. Assume that the two clauses are standardized apart. Suppose R is an OI-resolvent of C_1 and C_2, resolved upon the subsets of literals M and N, and let θ be the mgu$_{\text{OI}}(M, \overline{N})$ employed. Suppose the OI-interpretation I with domain $\mathcal{D}$ is an OI-model for $\{C_1, C_2\}$.
Let $\{X_1, \ldots, X_n\} = vars(C_1) \cup vars(C_2)$ and V be a variable assignment under object identity. Then, $\forall i = 1, \ldots, n,\ d_i \in \mathcal{D}$ if $M\theta$ is false under I and $V(X_1/d_1)\cdots(X_n/d_n)$, at least one of the other literals in C_1 is true under I and $V(X_1/d_1)\cdots(X_n/d_n)$. Similarly for $\overline{N}$. Note that R is made up of all literals in $C_1\theta$ and $C_2\theta$ except $M\theta$ or $N\theta$.
Since either $M\theta$ or $N\theta$ is false under I and $V(X_1/d_1)\cdots(X_n/d_n)$, at least one of the literals in R is true under I and $V(X_1/d_1)\cdots(X_n/d_n)$, hence I is an OI-model of R. Therefore, $\{C_1, C_2\} \models_{\text{OI}} R$.

This result can be inductively extended to OI-derivations, proving the soundness of OI-resolution:

Theorem 2.5 (Soundness) *Let Σ be a set of clauses, and C be a clause. If $\Sigma \vdash_{\text{OI}} C$ then $\Sigma \models_{\text{OI}} C$.*

Proof. Suppose $\Sigma \vdash_{\text{OI}} C$. Then there exists an OI-derivation $R_1, \ldots, R_k = C$ from Σ. By induction on k, it holds:

$(k = 1)$ *$R_1 = C \in \Sigma$, thus obviously $\Sigma \models_{\text{OI}} C$.*

$(k > 1)$ *Suppose the thesis holds for $m \leq k$.*
Let $R_1, \ldots, R_{k+1} = C$ be an OI-derivation of C from Σ. If $R_{k+1} \in \Sigma$ then the theorem is obvious. Otherwise, R_{k+1} is an OI-resolvent of some R_i and R_j $(i, j \leq k)$. By the inductive hypothesis, we have $\Sigma \models_{\text{OI}} R_i$ and $\Sigma \models_{\text{OI}} R_j$. From Lemma 2.4, it follows that $\{R_i, R_j\} \models_{\text{OI}} R_{k+1} = C$.

Now we can give a proof-theoretic definition of the notion of OI-implication. In the next section, we will justify this formally, showing the equivalence of the two versions.

Definition 2.10 *Let C and D be two clauses. Then C* implies *D* under object identity *(C* OI-implies *D), denoted $C \Rightarrow_{\text{OI}} D$ iff either D is a tautological clause or there exists a clause $E \in \mathcal{R}^*_{\text{OI}}(\{C\})$ such that E θ_{OI}-subsumes D. Equivalence under OI-implication is denoted by $\Leftrightarrow_{\text{OI}}$.*

From the definition above, it is easy to see that OI-implication is a strictly a stronger ordering relationship than θ_{OI}-subsumption. Moreover the set of most specific clauses $\perp$ will be made up by tautologies.

We will prove in the next section that Definitions 2.4 and 2.10 are equivalent, by means of a subsumption theorem.

3 Refutation Completeness

The soundness theorem for OI-implication bridges the gap from the proof-theoretic to the model-theoretic definition of this notion. In this section we give a proof of a completeness result to make the same step in the other direction. Following the proof-schema given in [Nienhuys-Cheng and de Wolf, 1996] for the standard case, first we prove this result for ground clauses, then we lift it to the general case.

3.1 Preliminaries

Let us start from the simplest case when both Σ and C are ground.

Lemma 3.1 *Let Σ be a logic program of ground clauses and C be a ground clause. If $\Sigma \models_{\text{OI}} C$ then there exists D such that $\Sigma \vdash_{\text{OI}} D$ and $D \subseteq C$.*

The proof is almost the same as in the case of standard implication (see [Nienhuys-Cheng and de Wolf, 1996]) since no variable assignment is made, for the clauses are ground. Yet in our case we considered finite sets of clauses. This can be extended by means of a compactness theorem.

In order to prove the Subsumption Theorem when only clause C is required to be ground, we need two theorems that are valid for the notion of unsatisfiability for standard interpretations but can be proven also when the object identity is assumed.

Theorem 3.2 *A set of clauses Σ is unsatisfiable under object identity iff there exists a finite unsatisfiable set Γ of ground instances of clauses from Σ.*

See [Nienhuys-Cheng and de Wolf, 1997] for a proof for standard interpretations. In this setting, it comes by using Proposition 2.3.

From this result and the Deduction Theorem another result follows:

Theorem 3.3 *Let Σ be a non-empty set of clauses and C be a ground clause. Then $\Sigma \models_{\text{OI}} C$ iff there exists a finite set Γ of ground instances of clauses from Σ such that $\Gamma \models_{\text{OI}} C$.*

Again, see [Nienhuys-Cheng and de Wolf, 1997] for a proof in the standard case. In this setting, Theorems 2.1 and 3.2 are to be exploited.

Now we have to show how to lift an OI-resolution step from the case of ground parent clauses to that of unrestricted clauses.

Lemma 3.4 *Let C_1 and C_2 be two clauses and C'_1 and C'_2, respectively, two instances of them. If R' is an OI-resolvent of C'_1 and C'_2, then there exists a clause R, that is an OI-resolvent of C_1 and C_2, such that R' is an instance of R.*

Proof. Suppose that C_1, C_2, C'_1 and C'_2 are standardized apart. Be $\sigma_{1,2}$ the OI-substitutions such that $C'_1 = C_1\sigma_1$ and $C'_2 = C_2\sigma_2$.
Let $N'_1 \subseteq C'_1$ and $N'_2 \subseteq C'_2$ be the subsets of literals resolved upon for obtaining R', then there exist $N_1 \subseteq C_1$ and $N_2 \subseteq C_2$ such that $N'_1 = N_1\sigma_1$ and $N'_2 = N_2\sigma_2$. Suppose that μ be an mgu$_{\text{OI}}(N'_1, \overline{N'_2})$.
Now let $D'_1 = C'_1 \setminus N'_1$ and $D'_2 = C'_2 \setminus N'_2$ for some $D_1 \subset N_1$ and $D_2 \subset N_2$. Hence, $R' = (D'_1 \cup D'_2)\mu = (D_1\sigma_1 \cup D_2\sigma_2)\mu$. Since the clauses are standardized apart, we can write: $R' = (D_1 \cup D_2)\sigma_1\sigma_2\mu$.
Consider N_1 and N_2. They can be unified through the OI-unifier $\sigma_1\sigma_2\mu$. Then, let θ be an mgu$_{\text{OI}}(N_1, \overline{N_2})$, there exists

an OI-substitution δ such that $\sigma_1\sigma_2\mu = \theta\delta$.
Thus, C_1 and C_2 can be resolved under object identity upon N_1 and N_2 by using θ: $R = ((C_1 \setminus N_1) \cup (C_2 \setminus N_2))\theta = (D_1 \cup D_2)\theta$. Then $R\delta = (D_1 \cup D_2)\theta\delta = (D_1 \cup D_2)\sigma_1\sigma_2\mu = R'$. Hence $R' = R\delta$.

We can generalize the lifting technique to the case of OI-derivations, i.e. to multiple OI-resolution steps.

Lemma 3.5 *Let Σ be a finite set of clauses and Σ' a set of instances of clauses from Σ. Suppose that $R'_1, \ldots, R'_k$ be an OI-derivation of clause R'_k from Σ'. Then there exists an OI-derivation $R_1, \ldots, R_k$ of clause R_k from Σ such that R'_i is an instance of R_i, for each $i = 1, \ldots, k$.*

Proof. By induction on the length of the derivation k.

($k = 1$) *In this case $R'_1 \in \Sigma'$. But Σ' is made up of instances of the clauses in Σ. Hence, $\exists R_1 \in \Sigma$ such that R'_1 is an instance of R_1.*

($k > 1$) *By hypothesis, let us suppose the thesis proven up to a derivation of length m. We have to prove it for $k = m + 1$.*
Let $R'_1, \ldots, R'_{m+1}$ be an OI-derivation of R'_{m+1} from Σ', so that R'_{m+1} is the resolvent of two clauses in $\Sigma' \cup \{R'_1, \ldots, R'_m\}$.
By hypothesis, there exists an OI-derivation $R_1, \ldots, R_m$ of R_m from Σ such that $\forall i = 1, \ldots, m : R'_i$ is an instance of R_i. Hence, by Lemma 3.4, there exists R_{m+1} such that R'_{m+1} is one of its instances.

We have now all the instruments for proving the following result:

Lemma 3.6 *Let Σ be a finite set of clauses and C be a ground clause. If $\Sigma \models_{\text{OI}} C$ then there exists D such that $\Sigma \vdash_{\text{OI}} D$ and $D \subseteq C$.*

Proof. Assume that C is not a tautology. We look for a clause D fulfilling the thesis. From $\Sigma \models_{\text{OI}} C$ and Theorem 3.3, then there exists a finite set Σ_g such that each clause in Σ_g is a ground instance of a clause in Σ and $\Sigma_g \models_{\text{OI}} C$.
Then, from Lemma 3.1 there exists a clause D' such that $\Sigma_g \vdash_{\text{OI}} D'$ and $D' \subseteq C$. Let $R'_1, \ldots, R'_k = D'$ be an OI-derivation of D' from Σ_g.
By means of Lemma 3.5, the OI-derivation can be lifted to an OI-derivation $R_1, \ldots, R_k$ of R_k from Σ, where $D' = R'_k$ is an instance of R_k. Let $D = R_k$. Then $\Sigma \vdash_{\text{OI}} D$ and D θ_{OI}-subsumes C (since $D' = D\delta$ and $D' \subseteq C$).

3.2 Subsumption Theorem and Consequences

Preliminarily, we have to recall the notion of a particular substitution that will be used in the next proofs:

Definition 3.1 *Let Σ be a set of clauses and C be a clause. Let $\{X_1, \ldots, X_n\} = vars(C)$ and suppose we have a set of constants $\{a_1, \ldots, a_n\} \cap consts(\Sigma \cup C) = \emptyset$. Then $\sigma = \{X_1/a_1, \ldots, X_n/a_n\}$ is a* Skolem substitution *for C w.r.t. Σ.*

This notion is easily extensible to this setting, it is easy to see that Skolem substitutions are also OI-substitutions.

Now, given the results of the previous sections, we can prove the main result and some consequences.

Theorem 3.7 (Subsumption Theorem) *Let Σ be a finite set of clauses and C be a clause. Then $\Sigma \models_{\text{OI}} C$ iff there exists a clause D such that $\Sigma \vdash_{\text{OI}} D$ and D θ_{OI}-subsumes C.*

Proof.

($\Rightarrow$) *Assume that C is not a tautology. Let θ be a Skolem OI-substitution for C w.r.t. Σ. Then $C\theta$ is a ground clause which is not a tautology and $\Sigma \models_{\text{OI}} C\theta$.*
By Lemma 3.6, there is a clause D such that $\Sigma \vdash_{\text{OI}} D$ and D θ_{OI}-subsumes $C\theta$. Since D is derived from Σ, D cannot contain constants yielded by θ. Therefore θ maps variables of C to constants that are not in D.
Let δ be an OI-substitution such that D θ_{OI}-subsumes $C\theta$ and σ be the OI-substitution obtained by replacing each binding X_i/t_i in δ with X_i/a_i. Then $\delta = \sigma\theta$. Since θ only replaces the variables X_i by a_i $(1 \leq i \leq n)$, it follows that $D\sigma \subseteq C$. Then D θ_{OI}-subsumes C. Hence, the thesis holds.

($\Leftarrow$) *For the soundness of the OI-derivation (and θ_{OI}-subsumption).*

As a consequence of this theorem, similarly to the standard case, it is nearly straightforward to demonstrate some important results originally due to Gottlob [1987].

Definition 3.2 *A clause C is* ambivalent *when there are two literals $L_1, L_2 \in C$ with the same predicate symbol but opposite sign. Besides, if they unify then the clause C is called* recursive.

Of course only unification through OI-substitutions is taken into account here. Given these notions, it is possible to show when OI-implication and θ_{OI}-subsumption coincide:

Proposition 3.8 *Let C and D be clauses. If C is not recursive and D is not tautological, then $C \models_{\text{OI}} D$ iff C θ_{OI}-subsumes D.*

Proof. Assume that $C \models_{\text{OI}} D$. By Theorem 3.7 it holds that $\exists E \in \mathcal{R}^*_{\text{OI}}(\{C\})$: E θ_{OI}-subsumes D. Since C is not recursive $\{C\} = \mathcal{R}^*_{\text{OI}}(\{C\})$. Hence, C θ_{OI}-subsumes D.
The ($\Leftarrow$) part is obvious.

Now, given a clause C, we denote with C^+ and C^-, respectively, the set of its positive and negative literals.

Proposition 3.9 *Let C and D be clauses. If $C \models_{\text{OI}} D$ then C^+ θ_{OI}-subsumes D^+ and C^- θ_{OI}-subsumes D^-.*

Proof. Observe that C^+ θ_{OI}-subsumes C then $C^+ \models_{\text{OI}} C$. By the hypothesis, it holds: $C \models_{\text{OI}} D$, thus $C^+ \models_{\text{OI}} D$. Note that by construction C^+ and D^+ contain only positive literals, then C^+ cannot be OI-resolved with itself, and D^+ is not a tautology. By Proposition 3.8, it must hold that C^+ θ_{OI}-subsumes D.
Again, observing that C^+ is made up of positive literals only, one can conclude that C^+ θ_{OI}-subsumes D^+.
Analogously C^- θ_{OI}-subsumes D^-.

This property can be used for the proof of the decidability of the implication under object identity. Yet, this subject goes beyond the scope of this paper.

Finally, we state a sufficient condition for the equivalence to hold between OI-implication and θ_{OI}-subsumption:

Proposition 3.10 *Let C and D be clauses. If D is not ambivalent, then $C \models_{\mathrm{OI}} D$ iff C θ_{OI}-subsumes D.*

Proof. Assume $C \models_{\mathrm{OI}} D$ and D non ambivalent. Thus D cannot be a tautology.
By Proposition 3.9 it holds $C^+\theta \subseteq D^+$ and $C^-\delta \subseteq D^-$ for some OI-substitutions θ and δ.
Now, if C were recursive then there would be a literal $L = L\sigma_1 = L\sigma_2$ such that $L\sigma_1 \in C^+$ and $\neg L\sigma_2 \in C^-$, with σ_1 and σ_2 OI-unifiers. But then $L\sigma_1\theta \in D^+$ and $\neg L\sigma_2\delta \in D^-$, which does not hold since D is not ambivalent.
Thus C is not recursive and D is not tautological. By Proposition 3.8, it follows that C θ_{OI}-subsumes D.

The Subsumption Theorem showed that OI-resolution, together with θ_{OI}-subsumption, can derive the same conclusions drawn model-theoretically by implication under object identity. In standard implication, resolution is not deduction complete (some θ-subsumption steps could be needed), yet it might be complete with respect to an unsatisfiable set of clauses. We prove this result in the object identity setting.

Theorem 3.11 (Refutation Completeness) *Let Σ be a finite set of clauses. Then Σ is unsatisfiable under object identity iff $\Sigma \vdash_{\mathrm{OI}} \square$.*

Proof.

($\Rightarrow$) *Be Σ a finite set of clauses such that $\Sigma \models_{\mathrm{OI}} \square$. For the Subsumption Theorem 3.7, there exists D such that $\Sigma \vdash_{\mathrm{OI}} D$ and D θ_{OI}-subsumes $\square$. Then D must be the empty clause itself. Hence $\Sigma \vdash_{\mathrm{OI}} \square$*

($\Leftarrow$) *For the soundness of the OI-derivation (Theorem 2.5).*

All the theorems and propositions of the previous section can be extended to the case of an infinite set of formulas Σ if a Compactness Theorem like that for standard models [Boolos and Jeffrey, 1989] can be demonstrated. Yet, here we are interested in these issues concerning logic programs, hence finite sets of clauses only.

4 Conclusions and Further Work

A framework has been presented for the novel notion of implication under object identity. By assuming this bias over a clausal representation, we were able to define OI-implication both model-theoretically and proof-theoretically. The two definitions can be unified by exploiting the soundness of the OI-derivations and a subsumption theorem which is the basis for proving the refutation completeness of our procedure. This relationship, that is stronger than θ_{OI}-subsumption, induces a quasi-order over the clausal search space conferring different algebraic properties than standard implication.

We expect encouraging results on the complexity of learning/revising logic theories in such a search space. Hence, from a theoretical viewpoint, we are now concerned with the demonstration of the compactness and decidability of this new relationship. Another research stream will concern the algebraic properties of the resulting search space. In particular, algorithms for calculating minimal generalizations and specializations are to be developed.

In a more long term perspective, we aim at building an integrated framework for multistrategy learning that will benefit by these theoretical findings.

References

[Badea and Stanciu, 1999] L Badea and M. Stanciu. Refinement operators can be (weakly) perfect. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming - ILP99*, volume 1634 of *LNAI*, pages 21–32. Springer, 1999.

[Boolos and Jeffrey, 1989] G.S. Boolos and R.C. Jeffrey. *Computability and Logic*. Cambridge University Press, Cambridge, U.K., 3rd edition, 1989.

[Chang and Lee, 1973] C.L. Chang and R.C.T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, San Diego, CA, 1973.

[Esposito *et al.*, 2000] F. Esposito, N. Fanizzi, S. Ferilli, and G. Semeraro. Ideal refinement under object identity. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning - ICML2000*, pages 263–270, Palo Alto, CA, 2000. Morgan Kaufmann.

[Gottlob and Fermüller, 1993] G. Gottlob and C.G. Fermüller. Removing redundancy from a clause. *Artificial Intelligence*, 61:263–289, 1993.

[Gottlob, 1987] G. Gottlob. Subsumption and implication. *Information Processing Letters*, 24(2):109–111, 1987.

[Idestam-Almquist, 1995] P. Idestam-Almquist. Generalization of clauses under implication. *Journal of Artificial Intelligence Research*, 3:467–489, 1995.

[Khardon, 1999] R. Khardon. Learning function-free Horn expressions. *Machine Learning*, 37(3):241–275, December 1999.

[Lloyd, 1987] J.W. Lloyd. *Foundations of Logic Programming*. Springer, 2nd edition, 1987.

[Nienhuys-Cheng and de Wolf, 1996] S.-H. Nienhuys-Cheng and R. de Wolf. The subsumption theorem in inductive logic programming: Facts and fallacies. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 265–276. IOS Press, 1996.

[Nienhuys-Cheng and de Wolf, 1997] S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *LNAI*. Springer, 1997.

[Reiter, 1980] R. Reiter. Equality and domain closure in first order databases. *Journal of ACM*, 27:235–249, 1980.

[Robinson, 1965] J.A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.

[Schmidt-Schauss, 1988] M. Schmidt-Schauss. Implication of clauses is undecidable. *Theoretical Computer Science*, 59:287–296, 1988.

[Semeraro *et al.*, 1998] G. Semeraro, F. Esposito, D. Malerba, N. Fanizzi, and S. Ferilli. A logic framework for the incremental inductive synthesis of Datalog theories. In N.E. Fuchs, editor, *Proceedings of LOPSTR97*, volume 1463 of *LNCS*, pages 300–321. Springer, 1998.

The Levelwise Version Space Algorithm and its Application to Molecular Fragment Finding

Luc De Raedt and **Stefan Kramer**
Albert-Ludwigs-University Freiburg
Institute for Computer Science
Georges-Köhler-Allee Geb. 079
D-79110 Freiburg in Breisgau, Germany
{deraedt, skramer}@informatik.uni-freiburg.de

Abstract

A tight integration of Mitchell's version space algorithm with Agrawal *et al.*'s Apriori algorithm is presented. The algorithm can be used to generate patterns that satisfy a variety of constraints on data. Constraints that can be imposed on patterns include the generality relation among patterns and imposing a minimum or a maximum frequency on data sets of interest.

The theoretical framework is applied to an important application in chemo-informatics, i.e. that of finding fragments of interest within a given set of compounds. Fragments are linearly connected substructures of compounds. An implementation as well as preliminary experiments within the application are presented.

1 Introduction

Mannila and Toivonen [Mannila and Toivonen, 1997] formulate the general pattern discovery task as follows. Given a database r, a language $\mathcal{L}$ for expressing patterns, and a constraint q, find the theory of r with respect to $\mathcal{L}$ and q, i.e. $Th(\mathcal{L}, r, q) = \{\phi \in \mathcal{L} \mid q(r, \phi)\ is\ true\}$. Viewed in this way $Th(\mathcal{L}, r, q)$ contains all sentences within the pattern language considered that make the constraint q true. This formulation of pattern discovery is generic in that it makes abstraction of several specific tasks including the discovery of association rules, frequent patterns, inclusion dependencies, functional dependencies, frequent episodes, ... Also, efficient algorithms for solving these tasks are known (cf. [Mannila and Toivonen, 1997]).

So far the the type of constraint that has been considered is rather simple and typically relies on the frequency of patterns. In the past decade the data mining community spent a lot of effort to efficiently compute patterns having a minimum frequency, such as e.g. Apriori [Agrawal *et al.*, 1993]. In this paper, we will extend this popular data mining model by allowing the user to specify a variety of different constraints on the patterns of interest. The constraints that will be considered involve generality constraints on patterns, e.g. to specify that the patterns of interest are (resp. are not) more general than a specific pattern, as well as frequency constraints. Frequency constraints that are considered either impose a maximum or a minimum frequency on a data set of interest. These constraints can then be combined e.g. in order to discover all patterns that are more general than pattern x, have a minimum frequency m_1 on the dataset p, and a maximum frequency m_2 on data set n. The result is a flexible and declarative query language to specify the patterns of interest. From this point of view, the work also fits in the inductive database framework considered by researchers such as [Imielinski and Mannila, 1996; Han *et al.*, 1999; De Raedt, 2000].

The key problem in discovering theories that involve a conjunction of primitive constraints $c_1 \wedge ... \wedge c_n$ is to efficiently combine the solvers for the primitive constraints c_i. It is at this point where Mitchell's version space approach [Mitchell, 1982] is extremely useful. Indeed, each of the primitive constraints results in a version space. Consider e.g. the minimum frequency constraint. As shown by [Mannila and Toivonen, 1997], the minimum frequency constraint results in a space of solutions with the most general pattern $\top$ as the only element of the G-set and the BD^+ boundary as the S-set. Because this property holds for all primitive constraints, the space of solutions of the conjunction of constraints can also be specified as a version space. Moreover, it can – in principle – be computed using Hirsh's version space intersection method [Hirsh, 1994]. However, rather than applying Hirsh's framework, we will employ a tighter integration of version spaces with Apriori.

To demonstrate the relevance of level-wise version spaces, we present an implementation as well as experiments in the domain of *molecular fragment finding*. Molecular fragments are sequences of linearly connected atoms. They are useful and important for the induction of so-called Structure-Activity Relationships (SARs), which are statistical models that relate chemical structure to biological activity. The use of automatically derived fragments in SARs originates from the CASE/MultiCASE systems developed by [Rosenkranz *et al.*, 1999]. With more than 150 published references, the CASE/MultiCASE systems are the most extensively used SAR and predictive toxicology systems. Previous approaches in these areas are based on the "decomposition" of individual compounds: these methods generate *all* fragments occurring in a given single compound. In this regard, our contribution is a language that enables the formulation of complex queries

regarding fragments – users can specify precisely which fragments they are interested in. We also implemented an efficient solver to answer queries in this language. Thus, from the algorithmic point of view, it is no longer necessary to process the results of queries post-hoc.

Molecular fragment finding has also been studied within the context of inductive logic programming and knowledge discovery in databases. For instance, Warmr [Dehaspe and Toivonen, 1999] or the approach by Inokuchi *et al.* [Inokuchi *et al.*, 2000] have been used in this context. Warmr is a system discovering frequently succeeding Datalog queries, and thus is not restricted to fragments. The approach by Inokuchi *et al.* deals with arbitrary frequent subgraphs, and thus is not restricted to linear fragments. Both approaches differ in that their pattern domain is more expressive, but finding frequent patterns is likely to be more expensive and complex than for linear fragments. Another approach to fragment finding is bottom-up propositionalization [Kramer and Frank, 2000]. All of these approaches handle only minimum frequency thresholds.

In contrast to all these approaches, the presented work allows the specification of all sorts of constraints, for instance regarding generality or frequency. Also, for the first time, one can pose constraints on the maximum frequency of fragments, and not only on the minimum frequency.

Finally, we would like to stress that the integration with version spaces results in a very compact representation of the resulting solutions. Previous methods would typically output all patterns within the version space. This is interesting for understandability reasons and also for further learning with these fragments as features.

The paper is organised as follows : in Section 2, we introduce the molecular fragment finding task and the primitives for querying such fragments, in Section 3, we present the level-wise version space algorithm, in Section 4, we discuss some experiments in molecular fragment finding, and in Section 5, we conclude and touch upon related work.

2 Framework

2.1 Molecular fragment finding

The task to which we will apply our integrated version space - Apriori framework is that of finding all molecular fragments that satisfy a conjunction of constraints $c_1 \wedge ... \wedge c_n$.

A *molecular fragment* is defined as a sequence of linearly connected atoms. For instance, '$o-s-c$' is a fragment meaning: "an oxygen atom with a single bond to a sulfur atom with a single bond to a carbon atom". In such expressions 'c', 'n', 'cl', etc. denote elements, and '$-$' denotes a single bond, '=' a double bond, '#' a triple bond, and '$\sim$' an aromatic bond. As common in the literature, we only consider "heavy" (i.e., non-hydrogen) atoms in this paper.

We assume that the system is given a database of example compounds and that each of the example compounds in the database is described using a 2-D representation. The information given there consists of the elements of the atoms of a molecule and the bond orders (single, double, triple, aromatic). An example compound in such a representation is shown in Fig. 1.

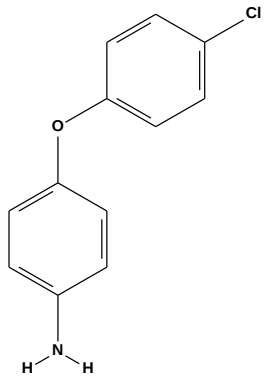


Figure 1: Example compound in a 2-D representation. '$cl - c \sim c \sim c \sim c - o$' is an example fragment occurring in the molecule.

A molecular fragment f *covers* an example compound e if and only if f considered as a graph is a subgraph of example e. For instance, fragment '$cl - c \sim c \sim c \sim c - o$' covers the example compound in Fig. 1.

There are a number of interesting properties of the language of molecular fragments $\mathcal{M}$:

- fragments in $\mathcal{M}$ are partially ordered by the *is more general than* relation; when fragment g is more general than fragment s we will write $g \leq s$;
- within this partial order, two syntactically different fragments are equivalent only when they are a reversal of one another; e.g. '$c - o - s$' and '$s - o - c$' denote the same substructure;
- $g \leq s$ if and only if g is a subsequence of s or g is a subsequence of the reversal of s; e.g. '$c-o$' $\leq$ '$c-o-s$'.
- there is a unique maximally general fragment (the empty fragment), which we denote by $\top$
- there is no maximally specific fragment; however, for convenience we add an artificial one to $\mathcal{M}$, which we denote by $\bot$.

Note that the representation of molecular fragments is relatively restricted compared to some other representations employed in data mining, such as first-order queries [Dehaspe and Toivonen, 1999] or subgraphs [Inokuchi *et al.*, 2000]. Although fragments are a relatively restricted representation of chemical structure, it is easy for trained chemists to recognize the functional group(s) that a given fragment occurs in. Thus, the interpretation of a fragment reveals more than meets the eye.

2.2 Constraints on fragments

The task addressed in this paper is that of finding the set of all fragments $f \in \mathcal{M}$ which satisfy a conjunction of primitive constraints $c_1 \wedge ... \wedge c_n$. The primitive constraints c_i imposed on the unknown target fragments f are :

- $f \leq p$, $p \leq f$, $\neg(f \leq p)$ and $\neg(p \leq f)$: where f is the unknown target fragment and p is a specific pattern; this type of primitive constraint denotes that f should (not)

be more specific (general) than the specified fragment p; e.g. the constraint '$c-o$' $\leq f$ specifies that f should be more specific than '$c-o$', i.e. that f should contain '$c-o$' as a subsequence;

- $freq(f, D)$ denotes the frequency of a fragment f on a set of molecules D; the frequency of a fragment f w.r.t. a dataset D is defined as the number of molecules in D that f covers;
- $freq(f, D_1) \leq t$, $freq(f, D_2) \geq t$ where t is a positive integer and D_1 and D_2 are sets of molecules; this constraint denotes that the frequency of f on the dataset D_i should be larger than (resp. smaller than) or equal to t; e.g. the constraint $freq(f, Pos) \geq 100$ denotes that the target fragments f should have a minimum frequency of 100 on the set of molecules Pos.

These primitive constraints can now conjunctively be combined in order to declaratively specify the target fragments of interest. Note that the conjunction may specify constraints w.r.t. any number of datasets, e.g. imposing a minimum frequency on a set of active molecules, and a maximum one on a set of inactive ones. E.g. the following constraint: $('c-o' \leq f) \wedge \neg(f \leq 'c-o-s-c-o-s') \wedge freq(f, Act) \geq 100 \wedge freq(f, InAct) \leq 5)$ queries for all fragments that include the sequence '$c-o$', are not a subsequence of '$c-o-s-c-o-s$', have a frequency on Act that is larger than 100 and and a frequency on $InAct$ that is smaller than 5.

3 Solving constraints

In this section, we will discuss how to find the set of all solutions $sol(c_1 \wedge ... \wedge c_n)$ in $\mathcal{M}$ to a conjunctive constraint $c_1 \wedge ... \wedge c_n$.

3.1 The search space

Due to the fact that the primitive constraints c_i are independent of one another, it follows that

$$sol(c_1 \wedge ... \wedge c_n) = sol(c_1) \cap ... \cap sol(c_n)$$

So, we can find the overall solutions by taking the intersection of the primitive ones.

Secondly, each of the primitive constraints c is monotonic or anti-monotonic w.r.t. generality (cf. [Mannila and Toivonen, 1997]). A constraint c is *anti-monotonic* (resp. monotonic) w.r.t. generality whenever

$$\forall s, g \in M : (g \leq s) \wedge (s \in sol(c)) \rightarrow (g \in sol(c))$$

(resp. $(g \in sol(c)) \rightarrow (s \in sol(c))$). The basic anti-monotonic constraints in our framework are: $(f \leq p), freq(f, D) \geq m$, the basic monotonic ones are $(p \leq f), freq(f, D) \leq m$. Furthermore the negation of a monotic constraint is anti-monotonic and vice versa.

Monotonic and anti-monotonic constraints are important because their solution space is bounded by a border. This fact is well-known in both the data mining literature (cf. [Mannila and Toivonen, 1997]), where the borders are often denoted by BD^+, as well as the machine learning literature (cf. [Mitchell, 1982]), where the symbols S and G are typically used.

To define borders, we need the notions of minimal and maximal elements of a set w.r.t. generality. Let F be a set of fragments, then define

$$max(F) = \{f \in F \mid \neg\exists q \in F : f \leq q\}$$
$$min(F) = \{f \in F \mid \neg\exists q \in F : q \leq f\}$$

We can now define the borders $S(c)$ and $G(c)$[1] of a primitive constraint c as

$$S(c) = min(sol(c)) \; and \; G(c) = max(sol(c))$$

Anti-monotonic constraints c will have $G(c) = \{\top\}$ and for proper constraints $S(c) \neq \{\bot\}$; proper monotonic constraints have $S(c) = \{\bot\}$ and $G(c) \neq \{\top\}$. Furthermore, as in Mitchell's version space framework we have that

$$sol(c) = \{f \in \mathcal{M} \mid \exists s \in S(c), \exists g \in G(c) : g \leq f \leq s\}$$

This last property implies that $S(c)$ (resp. $G(c)$) are proper borders for anti-monotone (resp. monotone) constraints.

So, we have that the set of solutions $sol(c_i)$ to each primitive constraint is a simple version space completely characterized by $S(c_i)$ and $G(c_i)$. Therefore, the set of solutions $sol(c_1 \wedge ... \wedge c_n)$ to a conjunctive constraint $c_1 \wedge ... \wedge c_n$ will also be completely characterized by the corresponding $S(c_1 \wedge ... \wedge c_n)$ and $G(c_1 \wedge ... \wedge c_n)$. At this point there are two issues:

1. computing the borders $S(c_i)$ and $G(c_i)$ for each primitive constraint c_i
2. computing the $S(c_1 \wedge ... \wedge c_n)$ and $G(c_1 \wedge ... \wedge c_n)$ given the individual borders $S(c_i)$ and $G(c_i)$

The first issue could be addressed using (variants of) the common level-wise algorithm for data mining (cf. [Mannila and Toivonen, 1997]). The second one could – in principle – be solved using Hirsh's version space merging algorithm. By now, it is probably clear that we need to integrate the level-wise algorithm with that of version spaces. However, rather than taking a loose coupling of the two approaches (as sketched above) we will provide a tighter integration of the two approaches. As we will show in the experimental section, this will lead to computational advantages.

The integrated algorithm computes the overall border sets incrementally. It initializes the borders S and G with the minimal and maximal elements, and then repeatedly updates for each primitive constraint. To update the borders with regard to a primitive constraint involving generality, it employs Mellish 's description identification algorithm. Mellish's algorithm extends Mitchell's version space algorithm in that it not only allows to process constraints fo the type $f \leq p$ and $\neg(f \leq p)$[2] but also handles the dual constraints $p \leq f$ and $\neg(p \leq f)$. Secondly, to update the version space w.r.t. frequency constraints, the integrated algorithm uses a variant of the level-wise algorithm that starts from the given borders rather than from $\top$ or $\bot$.

[1] At this point, we will follow Mitchell's terminology, because he works with two dual borders (a set of maximally general solutions G and a set of maximally specific ones S). In data mining, one typically only works with the S-set.

[2] The positive and negative examples in concept-learning.

3.2 Mellish's description identification algorithm

In order to formulate Mellish's description identification algorithm, we need to introduce some operations on fragments f_1 and f_2.

- $glb(f_1, f_2) = max\{f \mid f_1 \leq f \; and \; f_2 \leq f\}$
  the smallest "merged" fragments
- $lub(f_1, f_2) = min\{f \mid f \leq f_1 \; and \; f \leq f_2\}$
  the largest common subfragments
- $mgs(f_1, f_2) = max\{f \mid f_1 \leq f \; and \; not(f \leq f_2)\}$
  the smallest fragments more specific than f_1 but not more general than f_2
- $msg(f_1, f_2) = min\{f \mid f \leq f_1 \; and \; not(f_2 \leq f)\}$
  the largest fragments more general than f_1 but not more specific than f_2

Notice that these operators may generate more than one fragment; e.g. $lub('o = c = s', 'o = s') = \{'o', 's'\}$. These operations can now be used to instantiate Mellish's algorithm:

$S := \{\bot\}$; $G := \{\top\}$;
for all primitive constraint c **do**
  case c of $p \leq f$:
    $S := \{s \in S \mid p \leq s\}$
    $G :=$ max $\{glb \mid glb \in glb(p, g) \; and \; g \in G \; and$
      $\exists s \in S : glb \leq s\}$
  case c of $f \leq p$:
    $G := \{g \in G \mid g \leq p\}$
    $S :=$ min $\{lub \mid lub \in lub(p, s) \; and \; s \in S \; and$
      $\exists g \in G : g \leq lub\}$
  case i of $\neg(f \leq p)$:
    $S := \{s \in S \mid not(s \leq p)\}$
    $G :=$ max $\{m \mid \exists g \in G : m \in mgs(g, p)$
      $and \; \exists s \in S : m \leq s\}$
  case i of $\neg(p \leq f)$:
    $G := \{g \in G \mid not(p \leq g)\}$
    $S :=$ min $\{m \mid \exists s \in S : m \in msg(s, p)$
      $and \; \exists g \in G : g \leq m\}$

3.3 Variants of the level-wise algorithm

The algorithms outlined below employ refinement operators.

- A refinement operator $\rho_s(f) = max\{f' \in \mathcal{M} \mid f < f'\}$, i.e. extending a fragment by one atom.
- A generalization operator $\rho_g(P) = min\{f' \in \mathcal{M} \mid f' < P\}$, i.e. removing one atom from one side of a fragment.

To deal with the frequency constraints c, we may employ the following generalization of the level-wise algorithm. This is the *downwards* version.

Let c be a constraint of type $freq(f, D) \geq m$
$L_0 := G$; $i := 0$
while $L_i \neq \emptyset$ **do**
  $F_i := \{p \mid p \in L_i$ *and p satisfies constraint c*$\}$
  $I_i := L_i - F_i$ the set of infrequent fragments considered
  $L_{i+1} := \{p \mid \exists q \in L_i : p \in \rho_s(q)$
    $and \; \exists s \in S : p \leq s \; and \; \rho_g(p) \cap (\cup_j I_j) = \emptyset \}$
  $i := i + 1$
endwhile
$G := F_0$
$S := min(\cup_j F_j)$

To explain the algorithm, let us first consider the case where $S = \{\bot\}$ and $G = \{\top\}$. In this case, the above algorithm will behave roughly as the level-wise algorithm. The L_i will then contain only fragments of size i and the algorithm will keep track of the set of frequent fragments F_i as well as the infrequent ones. The algorithm will then repeatedly compute a set of candidate refinements L_{i+1}, delete those fragments that cannot be frequent by looking at the frequency of its generalizations, and evaluate the resulting possibly frequent fragments on the database. This process continues until L_i becomes empty.

The basic modifications to the level-wise algorithm that we made are concerned with the fact that we need not consider any fragment that is not in the already computed version space (i.e. any element not between an element of the G and the S set). Secondly, we have to compute the updated S set, which should contain all frequent fragments whose refinements are all infrequent.

Finding the updated G and S sets can also be realized in the dual manner. In this case, one will initialize L_0 with the elements of S and proceed otherwise completely dual.

The resulting *upwards* algorithm is shown below:

Let c be a constraint of type $freq(f, D) \geq m$
$L_0 := S$; $i := 0$
while $L_i \neq \emptyset$ **do**
  $F_i := \{p \mid p \in L_i$ *and p satisfies constraint c*$\}$
  $I_i := L_i - F_i$ the set of infrequent fragments considered
  $L_{i+1} := \{p \mid \exists q \in L_i : p \in \rho_g(q) \; and$
    $\exists g \in G : g \leq p \; and \; \rho_s(p) \cap (\cup_j F_j) = \emptyset \}$
  $i := i + 1$
endwhile
$G := \{g \in G \mid g \; satisfies \; c\}$
$S := min(\cup_j F_j)$

Whether the top down or bottom up version works more efficiently is likely to depend on the application and query under consideration. At this point it remains an open question as to when which strategy works more efficiently.

Finally, it is also possible to modify the above algorithms (exploiting the dualities) in order to handle *monotonic* frequency constraint of the form $freq(f, D) \leq m$. In this case, one can use the following algorithm (or its dual):

Let c be a constraint of type $freq(f, D) \leq m$
$L_0 := G$; $i := 0$
while $L_i \neq \emptyset$ **do**
  $I_i := \{p \mid p \in L_i$ *and p satisfies constraint c*$\}$
  $F_i := L_i - I_i$ the set of frequent fragments considered
  $L_{i+1} := \{p \mid \exists q \in L_i : p \in \rho_s(q)$
    $and \; \exists s \in S : p \leq s \; and \; \rho_g(p) \cap (\cup_j I_j) = \emptyset \}$
  $i := i + 1$
endwhile
$G := max(\cup_j I_j)$
$S := \{s \in S \mid s \; satisfies \; c\}$

3.4 Optimisations

Various optimisations to the algorithms are possible.

First, though we have adopted the standard level-wise algorithm to search for the borders when handling frequency constraints, it would also be possible to adopt some more recent and more efficient algorithms, such as those presented by [Bayardo, 1998; Gunopulos *et al.*, 1997]. These directly focus on the most specific (the longest) patterns, i.e. the S-set.

Secondly, Apriori-style algorithms can be made more efficient, if elements of one level in level-wise search are combined to give the candidates for the subsequent one. This can also be done for fragments. For instance, if '$o-s-c$' and '$s-c-o$' are known to be frequent at level 3, '$o-s-c-o$' is a candidate for a frequent fragment at level 4. However, several variants (with respect to order) have to be considered; e.g. '$o-s-c$' and '$s-o-c$' can be combined into '$c-s-o-c$' as well as into '$c-o-s-c$'.

Thirdly, we keep track of the fragments in canonical form. As indicated earlier, each fragment is equivalent to its reversal. In the implementation, we use the canonical form of a fragment which is defined as the maximum (w.r.t. a lexicographic ordering) of the fragment and its reversal. The implementation of the operators takes care of this.

Fourthly, one problem with the implementation of the framework for fragments stems from the fact that the bottom $\perp$ is not a "valid" fragment that can be manipulated. In particular, $\rho_g(\perp)$ is not defined. Thus, we cannot search upwards from the set S if $S = \{\perp\}$. Instead, we have to search downwards from G. If however, S is not equal to $\{\perp\}$, then we can process maximum frequency constraints "upwards" starting with S. For the same reason ($\rho_g(\perp)$ is undefined), we cannot start a query with a constraint $\neg(f \leq p)$, where p is a concrete fragment.

3.5 Optimisation primitives

Two primitives that seem especially useful are *minimize* and *maximize*. Indeed, one could imagine being interested in those fragments that satisfy a number of constraints and in addition have maximum frequency on a certain dataset or minimally general. It is easy to extend the framework with primitives $minimize(c, crit)$ and $maximize(c, crit)$ that finds those fragments in $\mathcal{M}$ that satisfy a conjunctive constraint c and that are minimal or maximal with regard to the specified criterion. In this paper we consider only criteria that are monotonic or anti-monotonic (such as frequency and generality).

In order to find the elements with regard to these optimisation primitives, one first computes the S and G sets with regard to c and then selects those elements within S or G (depending on the $crit$) that are minimal or maximal with regard to the criterion.

4 Experiments

In order to validate our approach, we applied it to the Predictive Toxicology Evaluation challenge dataset of Srinivasan et al. [Srinivasan *et al.*, 1999]. This data set consists of over 300 compounds (and takes more than 1 Mbyte of memory in its Prolog encoding) and has been used as a standard benchmark in predictive toxicology and artificial intelligence. In this application, the goal is to discover molecular fragments that are (relatively) frequent in carcinogenic compounds and infrequent in non-carcinogenic compounds. Such activating, toxic fragments are called *structural alerts* in the toxicological literature [Ashby and Patton, 1993]. One interesting question in this context is whether it is possible to rediscover known alerts. In the following, we summarize our experience with the new approach with example queries and systematic experiments.

4.1 Some interesting queries

One open research question in toxicological research is the role of chlorinated compounds in carcinogenicity. In the new framework, an example query concerning chlorinated fragments that are frequent in active compounds and infrequent in inactive ones looks as follows:

$$('cl' \leq f) \wedge (freq(f, Act) \geq 25) \wedge (freq(f, InAct) \leq 5)$$

A related query concerns the existence of activating, non-halogenated fragments:

$$\neg('f' \leq f) \wedge \neg('cl' \leq f) \wedge \neg('br' \leq f) \wedge \neg('i' \leq f) \wedge (freq(f, Act) \geq 25) \wedge (freq(f, InAct) \leq 5)$$

4.2 Quantitative results

To gather more quantitative evidence, we performed systematic experiments in the above domain. From the application side, it is quite clear that structural alerts are relatively rare for carcinogenicity, so that it can safely be assumed that alerts have a frequency of less than 25 in the positive, active compounds. Also, we are not interested in fragments with *any* frequencies in the positive resp. negative examples. Rather, we are seeking fragments that are, statistically significant, over-represented in the active compounds and under-represented in the inactives. Setting the minimum frequency in the actives to 6, 10, 16 and 20, respectively, we apply the χ^2-Test to a 2×2 contingency table with the class as one variable and the occurrence of the fragment as the other one to determine the maximum allowable frequency in the inactive compounds. In this way, we obtain maximum frequency thresholds of 0, 2, 5 and 7, respectively. For instance, we require the minimum to be 6 and the maximum to be 0 for the first experiment.

Methodwise, we performed a comparison of three approaches to the search for these fragments. Each of these approaches consists of two stages: the first stage handling the minimum frequency query (using the first algorithm in Section 3.3), and the second stage handling the maximum frequency query. The three approaches differ in the second stage, dealing with the maximum frequency query.

The first approach is based on version spaces, searching upwards from S (using the dual version of the third algorithm in Section 3.3). In Table 1, the results for this method can be found in the column for up. In contrast, the second one searches downwards, starting from $\top$ until all elements of set G are determined (using the third algorithm in Section 3.3 –

min	max	*up*	*down*	*post*
6	0	353.4	335.0	436.5
10	2	166.7	137.9	163.9
16	5	60.2	61.1	70.6
20	7	49.2	53.8	55.7

Table 1: Runtimes in seconds on a Pentium II.

column *down* in Table 1). The third method performs simple post-processing (column *post* in the table): it filters those fragments that are too frequent in the given dataset. Table 1 summarizes the runtimes of these three methods for the given minimum/maximum frequency parameter settings in seconds CPU time.

The outcome of these experiments is not clear a priori, because additional bookkeeping is done by our Version Space approach. Still, the experiments show that in 7 out of 8 cases, the Version Space approach pays: it outperforms the rather ad-hoc post-processing method in terms of computation time. Another result from the experiments is that – for the given queries – it does not make a big difference whether we search upwards or downwards for maximum frequency queries.

Perhaps the most important outcome of the experiments is the answers to the queries, which are shown below. They indicate that – for the given queries – version spaces constitute indeed a suitable and compact representation for the solution sets to the queries. Also, as outlined above, the computational time needed for answering the queries is reasonable.

```
 6:0: G={c-c-c-c-o-c-c~c}
      S={c-c-c-c-o-c-c~c~c~c~c~c}
10:2: G={c~c-c~c, br,
         c-o-c~c~c~c~c~c-n,c-o-c~c-n}
      S={c~c~c~c~c~c-c~c~c~c~c~c,
         br-c, c-o-c~c~c~c~c~c-n,
         c-o-c~c-n}
16:5: G=S={c-c~c~c~c-n}
20:7: G=S={n-c~c~c~c~c~c-o,
           c-c~c~c~c-n,n-c~c-o}
```

Further results and experiments in the context of feature construction and propositionalization can be found in [Kramer and De Raedt, 2001].

5 Conclusions and Related Work

The presented work contributes to 1) the theory of data mining and machine learning because of its integration of version spaces with the level-wise algorithm, 2) the framework of inductive databases, because the constraints can and should be interpreted as queries in a molecular fragment finding language, and, as discussed above, 3) to molecular fragment finding. We briefly review the key contributions in these domains and relate to relevant work where possible.

With regard to 1) and 2) our work builds on that by [De Raedt, 2000; 1998], who presents an integration of the version space and level-wise algorithms. However, we expand our earlier theoretical work in various respects. Indeed, in contrast to our earlier work, we report on an implementation, experiments that show the validity of the framework and an application in molecular fragment finding. Thus our work provides – for the first time – evidence that the framework is not only of theoretical interest but also effective with regard to applications.

With regard to 1), the presented algorithm provides a generalized theoretical framework for data mining. The resulting framework extends the borders in the levelwise techniques sketched by [Mannila and Toivonen, 1997], who link the level-wise algorithm to the S set of Mitchell's version space approach but do not further exploit the version space model. The experimental evidence indicates that this approach could form a viable extension to the classical level-wise algorithm.

Using two borders (i.e. the version space representation) to characterize the space of solutions to inductive queries is also done by [Dong and Li, 1999]. They use the version space representation to search for emerging patterns. Emerging patterns are defined as itemsets whose supports increase significantly from one dataset to another. Such patterns are also closely related to the significant fragments we discover using the χ^2 test. However, the primitive constraints we support seem to be different from those by [Dong and Li, 1999]. To compute the borders, Dong and Li do not employ the levelwise algorithm. Instead, they rely on more efficient algorithms such as Bayardo's [1998] Max-Miner. In principle it must be possible to adapt these more recent algorithms [Bayardo, 1998; Gunopulos *et al.*, 1997] to our framework too.

For what concerns 2), our work can also be regarded as a domain specific inductive database [Imielinski and Mannila, 1996; Meo *et al.*, 1998]. As sketched by [Han *et al.*, 1999], inductive databases allow the user to specify constraints on the patterns of interest. Recently, many of these constraints have been considered in the data mining literature, cf. e.g. [Ng *et al.*, 1998; Han *et al.*, 2000; 1999]. In this context however, the use of frequency constraints on different data sets seems new.

Finally, let us also note that the presented work on discovering molecular structures and regularities (also using version spaces) is related to the well-known Meta-Dendral system by [Buchanan and Mitchell, 1978].

References

[Agrawal *et al.*, 1993] R. Agrawal, T. Imielinski, A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Conference on Management of Data*, 1993.

[Ashby and Patton, 1993] J. Ashby and D. Paton. The Influence of Chemical Structure on the Extent and Sites of Carcinogenesis for 522 Rodent Carcinogens and 55 Different Human Carcinogen Exposures. *Mutation Research*, 286:3-74, 1993.

[Bayardo, 1998] R. Bayardo. Efficiently mining long patterns from databases. In *Proceedings of ACM SIGMOD Conference on Management of Data*, 1998.

[Buchanan and Mitchell, 1978] B. Buchanan and T. Mitchell. Model-directed learning of production rules. In Waterman, D. A. and Hayes-Roth, F. (Eds.) *Pattern-Directed Inference Systems*, Academic Press, New York, 1978.

[Dehaspe and Toivonen, 1999] L. Dehaspe, H. Toivonen. Discovery of Frequent Datalog Patterns, in *Data Mining and Knowledge Discovery Journal*, Vol. 3, 1999.

[De Raedt, 1998] L. De Raedt. An inductive logic programming language for database mining. in *Proceedings of the 4th International Conference on Artificial Intelligence and Symbolic Computation*, Lecture Notes in Artificial Intelligence, Vol. 1476, Springer Verlag, 2000.

[De Raedt, 2000] L. De Raedt. A Logical Database Mining Query Language. in *Proceedings of the 10th Inductive Logic Programming Conference*, Lecture Notes in Artificial Intelligence, Vol. 1866, Springer Verlag, 2000.

[Dong and Li, 1999] G. Dong and J. Li. Efficient mining of emerging patterns : discovering trends and differences. In *Proceedings of KDD*, ACM, 1999.

[Gunopulos *et al.*, 1997] D. Gunopulos, H. Mannila, S. Saluja: Discovering All Most Specific Sentences by Randomized Algorithms. In Foto N. Afrati, Phokion Kolaitis (Eds.): *Database Theory - ICDT '97, 6th International Conference*, Lecture Notes in Computer Science 1186, Springer 1997.

[Han *et al.*, 1999] J. Han, L. V. S. Lakshmanan, and R. T. Ng, Constraint-Based, Multidimensional Data Mining, *Computer*, Vol. 32(8): 46-50, 1999.

[Han *et al.*, 2000] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of ACM SIGMOD Conference on Management of Data* , 2000.

[Hirsh, 1994] H. Hirsh. Generalizing Version Spaces. *Machine Learning*, Vol. 17(1): 5-46 (1994).

[Imielinski and Mannila, 1996] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.

[Inokuchi *et al.*, 2000] A. Inokuchi, T. Washio, H. Motoda. An Apriori-based algorithm for mining frequent substructures from graph data. in D. Zighed, J. Komorowski, and J. Zyktow (Eds.) *Proceedings of PKDD 2000*, Lecture Notes in Artificial Intelligence, Vol. 1910, Springer-Verlag, 2000.

[Kramer and Frank, 2000] S. Kramer and E. Frank. Bottom-Up Propositionalization. *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, 156-162, 2000.

[Kramer and De Raedt, 2001] S. Kramer and L. De Raedt. Feature construction with version spaces for biochemical applications. In *Proceedings of the 18th International Conference on Machine Learning*, Morgan Kaufmann, 2001.

[Mannila and Toivonen, 1997] H. Mannila and H. Toivonen, Levelwise search and borders of theories in knowledge discovery, *Data Mining and Knowledge Discovery*, Vol. 1, 1997.

[Meo *et al.*, 1998] R. Meo, G. Psaila and S. Ceri, An extension to SQL for mining association rules. *Data Mining and Knowledge Discovery*, Vol. 2, 1998.

[Mellish, 1990] C. Mellish. The description identification algorithm. *Artificial Intelligence*, 1990.

[Mitchell, 1982] T. Mitchell. Generalization as Search, *Artificial Intelligence*, 1980.

[Ng *et al.*, 1998] R. T. Ng, L. V.S. Lkshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proceedings of ACM SIGMOD Conference on Management of Data* , 1998.

[Rosenkranz *et al.*, 1999] H.S. Rosenkranz, A.R. Cunningham, Y.P. Zhang, H.G. Clayhamp, O.T. Macina, N.B. Sussmann, S.G. Grant and G. Klopman. Development, Characterization and Application of Predictive-Toxicology Models. *SAR and QSAR in Environmental Research*, 10:277-298, 1999.

[Srinivasan *et al.*, 1999] A. Srinivasan, R.D. King and D.W. Bristol. An Assessment of Submissions Made to the Predictive Toxicology Evaluation Challenge. *Proc. of IJCAI-99*, 270-275, 1999.

MACHINE LEARNING AND DATA MINING

PROBABILISTIC LEARNING

Active Learning for Structure in Bayesian Networks

Simon Tong
Computer Science Department
Stanford University
simon.tong@cs.stanford.edu

Daphne Koller
Computer Science Department
Stanford University
koller@cs.stanford.edu

Abstract

The task of causal structure discovery from empirical data is a fundamental problem in many areas. Experimental data is crucial for accomplishing this task. However, experiments are typically expensive, and must be selected with great care. This paper uses *active learning* to determine the experiments that are most informative towards uncovering the underlying structure. We formalize the causal learning task as that of learning the structure of a causal Bayesian network. We consider an active learner that is allowed to conduct experiments, where it *intervenes* in the domain by setting the values of certain variables. We provide a theoretical framework for the active learning problem, and an algorithm that actively chooses the experiments to perform based on the model learned so far. Experimental results show that active learning can substantially reduce the number of observations required to determine the structure of a domain.

1 Introduction

Determining the causal structure of a domain is frequently a key issue in many situations. *Bayesian networks (BNs)* [Pearl, 1988] are a compact graphical representation of joint probability distributions. They can also be viewed as providing a causal model of a domain [Pearl, 2000]. If we assume that the graphical structure of the BN represents the causal structure of the domain, we can formalize the problem of discovering the causal structure of the domain as the task of learning the BN structure from data.

Over the last few years, there has been substantial work on discovering BN structure from purely observational data. However, there are inherent limitations on our ability to discover the structure based on randomly sampled data. Experimental data, where we intervene in the model, is vital for a full determination of the causal structure. However, obtaining experimental data is often time consuming and costly. Thus the experiments must be chosen with care.

In this paper, we provide an *active learning* algorithm that selects experiments that are most informative towards revealing the causal structure. With active learning the choice of the next data case is based upon the results seen so far. The possibility of active learning can arise naturally in a variety of domains and in several variants. In *interventional* active learning, the learner can ask for experiments involving interventions to be performed. This type of active learning is the norm in scientific studies: we can ask for a rat to be fed one sort of food or another. An intervention in the model causes certain probabilistic dependencies in the model to be replaced by our intervention [Pearl, 2000] — the rat no longer eats what it would normally eat, but what we choose it to. By observing the results of this experiment, we can determine the direction of causal influence in cases where purely observational data is inadequate.

In such active learning settings, where we have the ability to actively select experiments, we need a mechanism that tells us which experiment to perform next. We present a formal framework for active learning in Bayesian networks, based on the principles of Bayesian learning. We maintain a distribution over Bayesian network structures, which is updated based on our data. We define a notion of *quality* of our distribution, and provide an algorithm that selects queries in a greedy way, designed to improve model quality as much as possible. We provide experimental results on a variety of domains, showing that our active learning algorithm can provide substantially more accurate estimates of the BN structure using the same amount of data. Interestingly, our active learning algorithm provides significant improvements even in cases where it cannot intervene in the model, but only select instances of certain types. Thus, it is applicable even to the problem of learning structure in a non-causal setting.

2 Learning Bayesian Networks

Let $\mathcal{X} = \{X_1, \ldots, X_n\}$ be a set of random variables, with each variable X_i taking values in some finite domain *Dom*$[X_i]$. A *Bayesian network (BN)* over $\mathcal{X}$ is a pair $(G, \boldsymbol{\theta}_G)$ that represents a distribution over the joint space of $\mathcal{X}$. G is a directed acyclic graph, whose nodes correspond to the random variables in $\mathcal{X}$ and whose structure encodes conditional independence properties about the joint distribution. We use $\mathbf{U}_i$ to denote the set of parents of X_i. $\boldsymbol{\theta}_G$ is a set of parameters which quantify the network by specifying the *conditional probability distributions (CPDs)* $P(X_i \mid \mathbf{U}_i)$.

The Bayesian network represents a joint distribution over the set of variables $\mathcal{X}$ via the *chain rule for Bayesian networks*: $P(X_1, \ldots, X_n) = \prod_i P(X_i \mid \mathbf{U}_i)$. Viewed as a probabilistic model, it can answer any query of the form $P(\mathbf{Y} \mid \mathbf{Z} = \mathbf{z})$ where $\mathbf{Y}$ and $\mathbf{Z}$ are sets of variables and $\mathbf{z}$ an assignment of values to $\mathbf{Z}$. However, a BN can also be viewed as a *causal model* [Pearl, 2000]. Under this perspective, the BN can also be used to answer *interventional queries*, which

specify probabilities after we intervene in the model, forcibly setting one or more variables to take on particular values. In Pearl's framework, an intervention in a causal model that sets a single node $X := x$ replaces the standard causal mechanism of X with one where X is forced to take the value x. In graphical terms, this intervention corresponds to *mutilating* the model G by cutting the incoming edges to X. Intuitively, in the new model, X does not depend on its parents; whereas in the original model, the fact that $X = x$ would give us information (via evidential reasoning) about X's parents, in the experiment, the fact that $X = x$ tells us nothing about the values of X's parents. For example, in a fault diagnosis model for a car, if we observe that the car battery is not charged, we might conclude evidentially that the alternator belt is possibly defective, but if we deliberately drain the battery, then the fact that it is empty obviously gives us no information about the alternator belt. Thus, if we set $\mathbf{X} := \mathbf{x}$, the resulting model is a distribution where we mutilate G to eliminate the incoming edges to nodes in $\mathbf{X}$, and set the CPDs of these nodes so that $\mathbf{X} = \mathbf{x}$ with probability 1.

Our goal is to learn a BN structure from data. We assume that there are no hidden variables and in addition we make two further standard assumptions:

- **Causal Markov assumption:** The data is generated from an underlying Bayesian network $(G^*, \boldsymbol{\theta}^*)$ over $\mathcal{X}$.
- **Faithfulness assumption**: the distribution P^* over $\mathcal{X}$ induced by $(G^*, \boldsymbol{\theta}^*)$ satisfies no independences beyond those implied by the structure of G^*.

Our goal is to reconstruct G^* from the data. Clearly, given enough data, we can reconstruct P^*. However, in general, P^* does not uniquely determine G. For example, if our network G^* has the form $X \rightarrow Y$, then $Y \rightarrow X$ is equally consistent with P^*. Given only samples from P^*, the best we can hope for is to identify the *Markov equivalence class* [Pearl, 1988] of G: a set of network structures that induce precisely the same independence assumptions. In a Markov equivalence class, the skeleton of the network — the set connected (X, Y) pairs — is fixed; for some of the pairs, the direction of the edge is fixed, while the other edges can be directed either way [Spirtes *et al.*, 1993].

If we are given experimental as well as observational data, our ability to identify the structure is much larger [Cooper and Yoo, 1999]. Intuitively, assume we are trying to determine the direction of an edge between X and Y. If we are provided experimental data that intervenes at X, and we see that the distribution over Y does not change, while intervening at Y does change the distribution over X, we can conclude (based on the assumptions above) that the edge is $Y \rightarrow X$.

3 Bayesian learning with experimental data

As discussed in the introduction, our goal is to use active learning to learn the BN structure — learning from data where we are allowed to control certain variables by intervening at their values. We formalize this idea by assuming that some subset $\mathbf{Q}$ of the variables are *query variables*. The learner can select a particular instantiation $\mathbf{q}$ for $\mathbf{Q}$. The request $\mathbf{Q} := \mathbf{q}$ is called a *query*. The result of such a query is called the *response* and it is a randomly sampled instance $\mathbf{x}$ of all the *non-query* variables, conditioned on $\mathbf{Q} := \mathbf{q}$. In other words, $\mathbf{x}$ is the result of an experiment where we intervened in the model by setting $\mathbf{Q}$ to take the values $\mathbf{q}$; our assumptions then imply that $\mathbf{x}$ is sampled from the mutilated model described above.

We use a Bayesian framework to learn the BN structure. More precisely, we maintain a distribution over possible structures and their associated parameters. We begin with a prior over structures and parameters, and use Bayesian conditioning to update it as new data is obtained. Following [Heckerman *et al.*, 1995], we make several standard assumptions about the prior:

- **Structure Modularity**: The prior $P(G)$ can be written in the form: $P(G) = \prod_i P(Pa(X_i) = \mathbf{U}^G)$.
- **Parameter Independence**: $p(\boldsymbol{\theta}_G \mid G) = \prod_i p(\boldsymbol{\theta}_{X_i|\mathbf{U}_i} \mid G)$
- **Parameter Modularity**: For two graphs G and G', if $\mathbf{U}_i^G = \mathbf{U}_i^{G'}$ then: $p(\boldsymbol{\theta}_{X_i|\mathbf{U}_i^G} \mid G) = p(\boldsymbol{\theta}_{X_i|\mathbf{U}_i^{G'}} \mid G')$.

In this paper, we also assume that the CPD parameters are multinomials and that the associated parameter distributions are the conjugate Dirichlet distributions. However, our analysis holds for any distribution satisfying the parameter modularity assumption.

Given a complete, randomly sampled instance $\mathbf{d}$ over $\mathcal{X}$, using Bayes rule we have that the posterior distribution over G is proportional to: $P(\mathbf{d} \mid G)P(G)$. $P(\mathbf{d} \mid G)$ is the *marginal likelihood* of the data and can be expressed as an integral over all possible parameter values in G:

$$P(\mathbf{d} \mid G) = \int P(\mathbf{d} \mid G, \boldsymbol{\theta}_G) p(\boldsymbol{\theta}_G \mid G)\, d\boldsymbol{\theta}_G$$

Now, instead of having a complete random sample, suppose that we have an interventional query $\mathbf{Q} := \mathbf{q}$, and resulting response $\mathbf{x}$. We need to define how to update the distribution $P(G, \boldsymbol{\theta}_G)$ given this query and response. We break this into two problems by using the identity: $P(G, \boldsymbol{\theta}_G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}) = p(\boldsymbol{\theta}_G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, G) \cdot P(G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$. Thus we need to determine how to update the parameter density of a structure and also how to update the distribution over structures themselves.

For the first term in this expression, consider a particular network structure G and a prior distribution $p(\boldsymbol{\theta}_G)$ over the parameters of G. It is clear that we cannot use the resulting complete instance to update the parameters of the nodes $\mathbf{Q}$ themselves since we have forced $\mathbf{Q}$ to take on specific values. All other variables can, however, be updated with the complete instance. (Note that there is no issue of *selection bias* when updating the parameters of the ancestors of $\mathbf{Q}$ since all of incoming edges to query nodes $\mathbf{Q}$ are cut.) Thus, we define a variable Y to be *updateable in the context of an interventional query* $\mathbf{Q}$ if Y is not in $\mathbf{Q}$.

Our update rule for the parameter density is now very simple. Given a prior density $p(\boldsymbol{\theta}_G)$ and a response $\mathbf{x}$ from a query $\mathbf{Q} := \mathbf{q}$, we do standard Bayesian updating of the parameters, as in the case of randomly sampled instances, but we update only the Dirichlet distributions of updateable nodes. We use $p(\boldsymbol{\theta}_G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, G)$ to denote the distribution $p'(\boldsymbol{\theta}_G)$ obtained from this algorithm; this can be read as

"the density of $\boldsymbol{\theta}_G$ after performing query $\mathbf{q}$ and obtaining the complete response $\mathbf{x}$". Note that this is quite different from the density $p(\boldsymbol{\theta}_G \mid \mathbf{q}, \mathbf{x}, G)$ which denotes standard Bayesian conditioning. We also note that performing such an interventional update to the parameters still preserves parameter modularity.

Now consider the distribution over structures. We use $P(G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$ to denote the posterior distribution over structures after performing the query and obtaining the response. The following theorem tells us how we can easily update the posterior over G given an interventional query:

Theorem 3.1 *Given a query $\mathbf{Q} := \mathbf{q}$ and complete response $\mathbf{x}$, if $P(G, \boldsymbol{\theta}_G)$ satisfies* parameter independence *and* parameter modularity*, then:*

$$P(G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}) = \frac{P(G)}{P(\mathbf{x} \mid \mathbf{Q} := \mathbf{q})} \prod_{i: X_i \notin \mathbf{Q}} \mathit{Score}(X_i, \mathbf{U}_i^G \mid \mathbf{x}, \mathbf{q}),$$

with $\mathit{Score}(X_i, \mathbf{U} \mid \mathbf{d}) = \int P(x_i \mid \mathbf{u}, \boldsymbol{\theta}_{X_i|\mathbf{u}}) p(\boldsymbol{\theta}_{X_i|\mathbf{u}})\, d\boldsymbol{\theta}_{X_i|\mathbf{u}}$.

Proof outline: Note that $\mathit{Score}(X_i, \mathbf{U} \mid \mathbf{d}) = P(x_i \mid \mathbf{u})$ where x_i and $\mathbf{u}$ are the values of X_i and $\mathbf{U}$ in the data instance $\mathbf{d}$. We have that $\prod_{i: X_i \notin \mathbf{Q}} \mathit{Score}(X_i, \mathbf{U}_i^G \mid \mathbf{x}, \mathbf{q}) = P(\mathbf{x} \mid \mathbf{Q} := \mathbf{q}, G)$. Finally, notice that $P(G) = P(G \mid \mathbf{Q} := \mathbf{q})$. □

4 Active Learning

Our goal in this paper is not merely to update the distribution based on interventional data. We want to *actively select* instances that will allow us to learn the structure better.

A *(myopic) active learner* ℓ is a function that selects a query $\mathbf{Q} := \mathbf{q}$ based upon its current distribution over G and $\boldsymbol{\theta}_G$. It takes the resulting response $\mathbf{x}$, and uses it to update its distribution over G and $\boldsymbol{\theta}_G$. It then repeats the process. We described the update process in the previous section. Our task now is to construct an algorithm for deciding on our next query given our current distribution P.

4.1 Loss function

As in the work of Tong and Koller [2001], a key step in our approach is the definition of a measure for the quality of our distribution over graphs and parameters. We can then use this measure to evaluate the extent to which various instances would improve the quality of our distribution, thereby providing us with an approach for selecting the next query to perform.

More formally, given a distribution over graphs and parameters $P(G, \boldsymbol{\theta}_G)$ we have a *loss function* $\mathit{Loss}(P)$ that measures the quality of our distribution over the graphs and parameters. Given a query $\mathbf{Q} := \mathbf{q}$ we define the *expected posterior loss* of the query as:

$$\begin{aligned} &\mathit{ExPLoss}(P(G, \boldsymbol{\theta}_G) \mid \mathbf{Q} := \mathbf{q}) \\ &\quad = \quad E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q} := \mathbf{q})} \mathit{Loss}(P(G, \boldsymbol{\theta}_G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})). \quad (1) \end{aligned}$$

This definition immediately leads to the following simple algorithm: For each candidate query $\mathbf{Q} := \mathbf{q}$, we evaluate the expected posterior loss, and then select the query for which it is lowest. Note, however, that the expected loss appears to be computationally expensive to evaluate. We need to maintain a distribution over the set of structures, and the number of structures in $\mathcal{G}$ is super-exponential in the number of nodes. Furthermore, given a query, to compute the expected posterior loss we have to perform a computation over the set of structures for each of the exponential number of possible responses to the query.

To make this high-level framework concrete, we must pick a loss function. Recall that our goal is to learn the correct structure; hence, we are interested in the presence and direction of the edges in the graph. For two nodes X_i and X_j, there are three possible edge relationships between them: either $X_i \rightarrow X_j$, or $X_i \leftarrow X_j$ or $X_i \quad X_j$. Our distribution P over graphs and parameters induces a distribution over these three possible edge relationships. We can measure the extent to which we are sure about this relationship using the entropy of this induced distribution:

$$\begin{aligned} H(X_i \leftrightarrow X_j) = &-P(X_i \rightarrow X_j) \log P(X_i \rightarrow X_j) \\ &-P(X_i \leftarrow X_j) \log P(X_i \leftarrow X_j) \\ &-P(X_i \quad X_j) \log P(X_i \quad X_j) \quad (2) \end{aligned}$$

The larger this entropy, the less sure we are about the relationship between X_i and X_j. This expression forms the basis for our *edge entropy* loss function:

$$\mathit{Loss}(P(G, \boldsymbol{\theta}_G)) = \sum_{i,j} H(X_i \leftrightarrow X_j) \quad (3)$$

In certain domains we may be especially interest in determining the relationship between particular pairs of nodes. We can reflect this desire in our loss function by introducing scaling factors in front of different $H(X_i \leftrightarrow X_j)$ terms.

Now that we have defined the loss function for a distribution $P(G, \boldsymbol{\theta}_G)$, our task is to find an efficient algorithm for computing the expected posterior loss of a given query $\mathbf{Q} := \mathbf{q}$ relative to P. We note that P is our current distribution, conditioned on all the data obtained so far. Initially, it is the prior; as we get more data, we use Bayesian conditioning (as described above) to update P, and then apply the same algorithm to the posterior.

Our approach to obtaining a tractable algorithm is based on the ideas of Friedman and Koller [2000] — we first consider the simpler problem of restricting attention to network structures consistent with some total ordering, $\prec$. Then, we introduce a distribution over the orderings.

4.2 Analysis for a Fixed Ordering

Let $\prec$ be a total ordering of $\mathcal{X}$. We restrict attention to network structures that are consistent with $\prec$, i.e., if there is an edge $X \rightarrow Y$, then $X \prec Y$. Following [Friedman *et al.*, 1999], we also assume that each node X_i has a set $\mathbf{W}_i$ of at most m possible *candidate* parents that is fixed before each query round. In certain domains, we can use prior knowledge to construct $\mathbf{W}_i$; in others we can use a technique mentioned in [Friedman and Koller, 2000] where we can use the data itself to point out nodes that are more likely to be directly related to X_i. We define the set of candidate parents for a node X_i that are consistent with our ordering as: $\mathcal{U}_{i,\prec} = \{\mathbf{U} : \mathbf{U} \prec X_i, \mathbf{U} \subseteq \mathbf{W}_i\}$, where $\mathbf{U} \prec X_i$ if $Y \prec X_i$ for all $Y \in \mathbf{U}$. We note that the number of structures in

induced by $\prec$ and $\mathbf{W}_i$ is still exponential in the number of variables in $\mathcal{X}$.

The key impact of the restriction to a fixed ordering is that the choice of parents for one node is independent of the choice of parents for another node [Buntine, 1991; Friedman and Koller, 2000]. Two important consequences are the following theorems [1] , which give us closed form, efficiently computable expressions for key quantities:

Theorem 4.1 *Given a query $\mathbf{Q} := \mathbf{q}$, we can write the probability of a response $\mathbf{x}$ to our query as:*

$$P(\mathbf{x} \mid \mathbf{Q} := \mathbf{q}, \prec)$$
$$= \sum_{G \in \prec} \prod_i P(Pa(X_i) = \mathbf{U}_i^G) \prod_{j: X_j \notin \mathbf{Q}} Score(X_j, \mathbf{U}_j^G \mid \mathbf{x}, \mathbf{q})$$
$$= \lambda_{\mathbf{Q}} \prod_{i: X_i \notin \mathbf{Q}} \sum_{\mathbf{U} \in \mathcal{U}_{i,\prec}} P(Pa(X_i) = \mathbf{U}) Score(X_i, \mathbf{U} \mid \mathbf{x}, \mathbf{q}),$$

where $\lambda_{\mathbf{Q}} = \prod_{i: X_i \in \mathbf{Q}} \sum_{\mathbf{U} \in \mathcal{U}_{i,\prec}} P(Pa(X_i) = \mathbf{U})$.

Theorem 4.2 *Given a query $\mathbf{Q} := \mathbf{q}$ and completion $\mathbf{x}$ we can write the probability of an edge $X_j \rightarrow X_i$ as:*

$$P(X_j \rightarrow X_i \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, \prec) = \frac{\sum_{\mathbf{U} \in \mathcal{U}_{i,\prec}, \mathbf{U} \ni X_j} P(Pa(X_i) = \mathbf{U}) Score(X_i, \mathbf{U} \mid \mathbf{x}, \mathbf{q})}{\sum_{\mathbf{U} \in \mathcal{U}_{i,\prec}} P(Pa(X_i) = \mathbf{U}) Score(X_i, \mathbf{U} \mid \mathbf{x}, \mathbf{q})},$$

where we define $Score(X_i, \mathbf{U} \mid \mathbf{x}, \mathbf{q}) = 1$ *if* $X_i \in \mathbf{Q}$.

Notice that since we are performing Bayesian averaging over multiple graphs the probability of an edge $X_i \rightarrow X_j$ will generally only be high if X_i is a *direct* cause of X_j rather than if X_i merely has some indirect causal influence on X_j.

Now, consider the expected posterior loss (Eq. (1)) given $\prec$:

$$ExPLoss_{\prec}(P(G, \boldsymbol{\theta}_G) \mid \mathbf{Q} := \mathbf{q})$$
$$= E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q} := \mathbf{q}, \prec)} \sum_{i,j} H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, \prec). \quad (4)$$

We can compute $H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, \prec)$ by using Theorem 4.2. Also, notice from Theorem 4.2 that the expression $H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, \prec)$ depends only on the values that $\mathbf{q}$ and $\mathbf{x}$ give to X_i, X_j, $\mathbf{W}_i$ and $\mathbf{W}_j$. Using this fact and then applying Theorem 4.1, we can rewrite the expected posterior loss as:

$$ExPLoss_{\prec}(P(G, \boldsymbol{\theta}_G) \mid \mathbf{Q} := \mathbf{q})$$
$$= E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q} := \mathbf{q}, \prec)} \sum_{i,j} H(X_i \leftrightarrow X_j \mid x_i, x_j, \mathbf{w}_i, \mathbf{w}_j, \prec)$$
$$= \sum_{i,j} \sum_{\mathbf{x}} P(\mathbf{x} \mid \mathbf{Q} := \mathbf{q}, \prec) H(X_i \leftrightarrow X_j \mid x_i, x_j, \mathbf{w}_i, \mathbf{w}_j, \prec)$$
$$= \sum_{i,j} \sum_{\mathbf{x}} H(X_i \leftrightarrow X_j \mid x_i, x_j, \mathbf{w}_i, \mathbf{w}_j, \prec) \times \lambda_{\mathbf{Q}} \prod_{k: X_k \notin \mathbf{Q}} \sum_{\mathbf{U} \in \mathcal{U}_{k,\prec}} P(Pa(X_k) = \mathbf{U}) Score(X_k, \mathbf{U} \mid \mathbf{x}, \mathbf{q})$$
$$= \lambda_{\mathbf{Q}} \sum_{i,j} \sum_{\mathbf{x}} \psi(x_i, x_j, \mathbf{w}_i, \mathbf{w}_j) \prod_{k: X_k \notin \mathbf{Q}} \phi(x_k, \mathbf{w}_k). \quad (5)$$

[1] Proofs are provided in the full version of this paper available at: robotics.stanford.edu/~stong/papers/tong_koller_ijcai01_full.ps

Where,

$$\psi(x_i, x_j, \mathbf{w}_i, \mathbf{w}_j) = H(X_i \leftrightarrow X_j \mid x_i, x_j, \mathbf{w}_i, \mathbf{w}_j, \prec)$$
$$\phi(x_k, \mathbf{w}_k) = \sum_{\mathbf{U} \in \mathcal{U}_{k,\prec}} P(Pa(X_k) = \mathbf{U}) Score(X_k, \mathbf{U} \mid x_k, \mathbf{w}_k).$$

This expression still involves summations over the exponential number of possible completions of a query. However, notice that for each i and j in Eq. (5), the summation over completions $\mathbf{x}$ resembles the expression for computing a marginal probability in Bayesian network inference where we are marginalizing out $\mathbf{x}$. In fact ψ and each ϕ can be regarded as factors and we can use standard graphical model inference procedures [Lauritzen and Spiegelhalter, 1988] to evaluate this expression effectively. The restriction to a candidate set of parents for each node ensures that each factor ϕ is over at most $(m+1)$ variables, and each factor ψ over at most $2(m+1)$ variables. After applying Bayesian network inference we end up with a factor over the variables $\mathbf{Q}$ where for each possible query $\mathbf{q}$ we have the value of the expression $\sum_{\mathbf{x}} \psi(x_i, x_j, \mathbf{w}_i, \mathbf{w}_j) \prod_{k: X_k \notin \mathbf{Q}} \phi(x_k, \mathbf{w}_k)$.

We need to perform such an inference for each i, j pair. However, since we restricted to at most m candidate parents, the number of possible edges is at most mn. Thus, the computational cost of computing the expected posterior loss for all possible queries is the cost of mn applications of Bayesian network inference.

4.3 Analysis for Unrestricted Orderings

In the previous section, we obtained a closed form expression for computing the expected posterior loss of a query for a given ordering. We now generalize this derivation by removing the restriction of a fixed ordering. The expression for the expected posterior loss can be rewritten as:

$$ExPLoss(P(G, \boldsymbol{\theta}_G) \mid \mathbf{Q} := \mathbf{q})$$
$$= E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q} := \mathbf{q})} Loss(P(G, \boldsymbol{\theta}_G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}))$$
$$= E_{\prec} E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q} := \mathbf{q}, \prec)} Loss(P(G, \boldsymbol{\theta}_G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}))$$
$$= E_{\prec} E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q} := \mathbf{q}, \prec)} \sum_{i,j} H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}).$$

The expectation over orderings can be approximated by sampling possible orderings from our current distribution over graphs and parameters. As shown by Friedman and Koller [2000], sampling from orderings can be done very effectively using Markov chain Monte Carlo (MCMC) techniques.

The expression inside the expectation over orderings is very similar to the expected posterior loss of the query with a fixed ordering (Eq. (4)). The only difference is that we now must compute the entropy terms $H(X_i \leftrightarrow X_j \mid \mathbf{x}, \mathbf{Q} := \mathbf{q})$ without restricting ourselves to a single ordering. This entropy term is based on probability expressions for relationships between nodes:

$$P(X_i \rightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}) = E_{\prec \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}} P(X_i \rightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, \prec). \quad (6)$$

Each of the terms inside the expectation can be computed using Theorem 4.2. Naively, we can compute the expectation for each query $\mathbf{Q} := \mathbf{q}$ and completion $\mathbf{x}$ by sampling orderings from $P(\prec \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$ and then computing $P(X_i \rightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, \prec)$. Clearly, this approach

is impractical. However, we can use a simple approximation that substantially reduces the computational cost. Our general MCMC algorithm generates a set of orderings sampled from $P(\prec)$. In many cases, a single data instance will only have a small effect on the distribution over orderings; hence, we can often use our samples from $P(\prec)$ to be a reasonably good approximation to samples from the distribution $P(\prec | \mathbf{Q} := \mathbf{q}, \mathbf{x})$. Thus, we use our current set of sampled orderings to approximate Eq. (6). Note that this small approximation error will not accumulate since we are not using the approximation to update any of the parameters of our model, just merely to predict the value of possible queries in this current round.

We note that, as in the fixed ordering case, the entropy term $H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$ depends only on the values given to the variables $X_i, X_j, \mathbf{W}_i$ and $\mathbf{W}_j$. Thus, we can use the same Bayesian network inference method to compute the expression $E_{\mathbf{x} \sim P(\mathbf{X} | \mathbf{Q}:=\mathbf{q}, \prec)} \sum_{i,j} H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$.

4.4 Algorithm Summary and Properties

To summarize the algorithm, we first sample a set of orderings from the current distribution over graphs and parameters. We then use this set of orderings to compute the entropy terms $H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$. Next, for each ordering we compute $E_{\mathbf{x} \sim P(\mathbf{X} | \mathbf{Q}:=\mathbf{q}, \prec)} \sum_{i,j} H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$ using a standard Bayesian network inference algorithm to obtain a factor over all possible queries. We then average all of these query factors obtained from each ordering. The final result is a query factor that, for each possible query, gives the expected posterior loss of asking that query. We then choose to ask the query that gives the lowest expected posterior loss.

We now consider the computational complexity of the algorithm. For each ordering we need to compute $E_{\mathbf{x} \sim P(\mathbf{X} | \mathbf{Q}:=\mathbf{q}, \prec)} \sum_{i,j} H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$. This involves at most mn Bayesian network inferences. Each inference returns a factor over all possible queries and so the inference will take time exponential in the number of query variables. The time complexity of our algorithm to generate the next query is: $\mathcal{O}$(# of sampled orderings $\cdot\ mn \cdot$ cost of BN inference).

In addition, we need to generate the sampled orderings themselves. Friedman and Koller [2000] provide techniques that greatly reduce the cost of this process. They also show that the Markov chain mixes fairly rapidly, thereby reducing the number of steps in the chain required to generate a random sample. In our setting, we can reduce the number of steps required even further. Initially, we start with a uniform prior over orderings, from which it is easy to generate random orderings. Each of these is now the starting point for a Markov chain. As we do a single query and get a response, the new posterior distribution over orderings is likely to be very similar to the previous one. Hence, our old set of orderings is likely to be be fairly close to the new stationary distribution. Thus, a very small number of MCMC steps from each of the current orderings will give us a new set of orderings which is very close to being sampled from the new posterior.

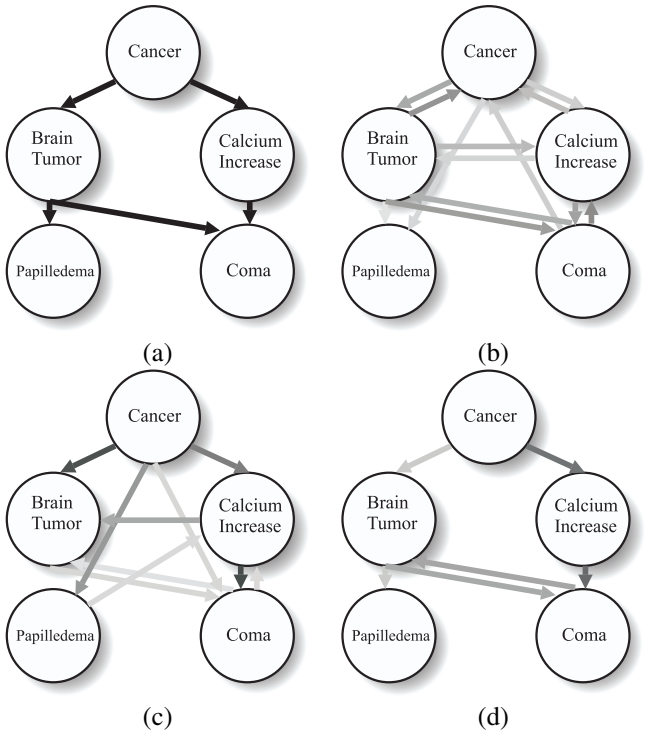


Figure 2: (a) Original **Cancer** network. (b) **Cancer** network after 70 observations. (c) **Cancer** network after 20 observations and 50 uniform experiments. (d) **Cancer** network after 20 observations and 50 active experiments. The darker the edges the higher the probability of edges existing. Edges with less than 15% probabilty are omitted to reduce clutter.

5 Experimental Results

We evaluated the ability of our algorithm to reconstruct the network structure by using data generated from a known network. We experimented with three commonly used networks: **Cancer**, with five nodes; **Asia**, with eight nodes; and **Car Troubleshooter**, with twelve nodes. For each test network, we maintained 50–75 orderings, as described above. We restricted the set of candidate parents to have size $m = 5$. It typically took a few minutes for the active method to generate the next query.

We compared our active learning method with both random sampling and uniform querying, where we choose a setting for the query nodes from a uniform distribution. Each method produces estimates for the probabilities of edges between each pair of variables in our domain. We compared each method's estimate with the true network G^* by using the L_1 *edge error* of the estimate:

$$\begin{aligned} \mathit{Error}(P) = \sum_{i,j>i} & I_{G^*}(X_i \rightarrow X_j)(1 - P(X_i \rightarrow X_j)) \\ & + I_{G^*}(X_i \leftarrow X_j)(1 - P(X_i \leftarrow X_j)) \\ & + I_{G^*}(X_i \quad X_j)(1 - P(X_i \quad X_j)) \end{aligned} \tag{7}$$

where $I_{G^*}(A) = 1$ if A holds in G^* and is zero otherwise.

We first considered whether the active method provides any benefit over random sampling other than the obvious additional power of having access to queries that intervene in the model. Thus, for the first set of experiments, we elimi-

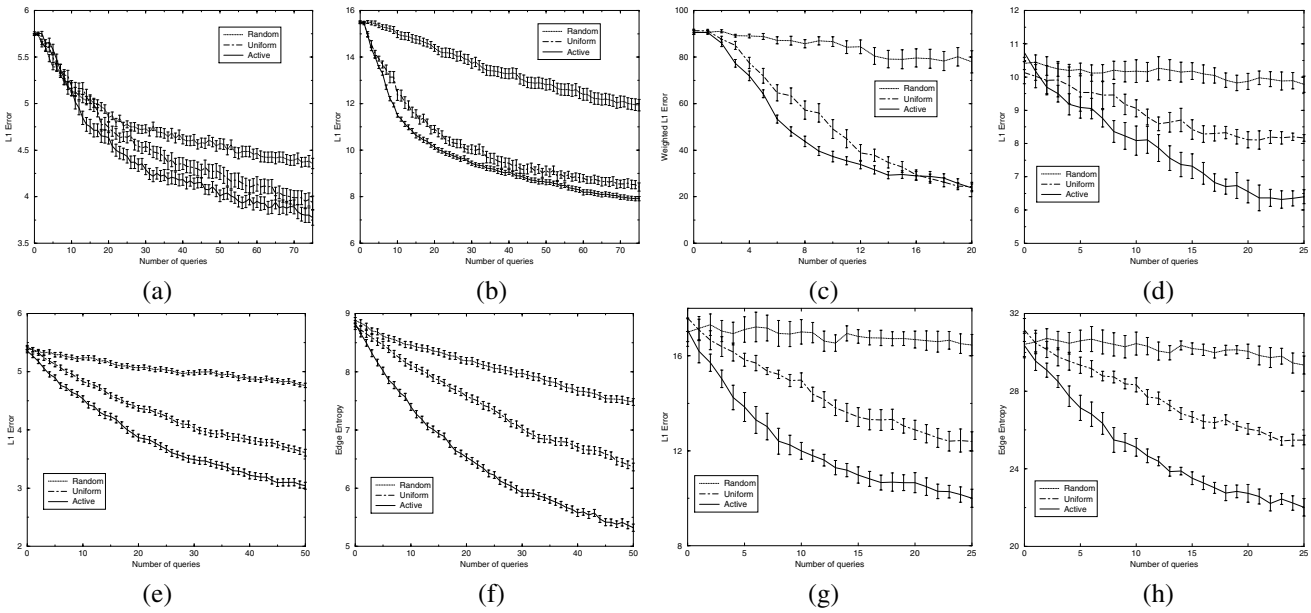


Figure 1: (a) **Cancer** with one query node. (b) **Car** with four query nodes. (c) **Car** with three query nodes and weighted edge importance. (d) **Asia** with any pairs or single or no nodes as queries. (e) **Cancer** with any pairs or single or no nodes as queries. (g) **Cancer** edge entropy. (f) **Car** with any pairs or single or no nodes as queries. (h) **Car** edge entropy. Legends reflect order in which curves appear. The axes are zoomed for resolution.

nated this advantage by restricting the active learning algorithm to query only roots of G^*. When the query is a root, a causal query is equivalent to simply selecting a data instance that matches the query (e.g., "Give me a 40-year-old male"); hence, there is no need for a causal intervention to create the response. Situations where we can only query root nodes arises in many domains; in medical domains, for example, we often have the ability to select subjects of a certain age, gender, or ethnicity, variables which are often assumed to be root nodes. All algorithms were informed that these nodes were roots by setting their candidate parent sets to be empty. In this batch of experiments, the candidate parents for the other nodes were selected at random, except that the node's true parents in the generating network were always in its candidate parent set.

Figures 1(a) and 1(b) show the learning curves for the **Cancer** and **Car** networks. We experimented with using uniform Dirichlet (BDe) priors and also more informed priors (simulated by sampling 20 data instances from the true network[2]). The type of prior made little qualitative difference in the comparative performance between the learning methods (the graphs shown are with uniform priors). In both graphs, we see that the active method performs significantly better than random sampling and uniform querying.

In some domains, determining the existence and direction of causal influence between two particular nodes may be of special importance. We experimented with this in the **Car** network. We modified the L1 edge error function Eq. (7) (and the edge entropy Eq. (3) used by the active method) to make determining the relationship between two particular nodes (the *FuelSubsystem* and *EngineStart* nodes) 100 times more important than a regular pair of nodes. We used three other nodes in the network as query nodes. The results are shown in Fig. 1(c). Again, the active learning method performs substantially better.

Note that, without true causal interventions, all methods have the same limited power to identify the model: asymptotically, they will identify the skeleton and the edges whose direction is forced in the Markov equivalence class (rather than identifying all edge directions in the true causal network). However, even in this setting, the active learning algorithm allows us to derive this information significantly faster.

Finally, we considered the ability of the active learning algorithm to exploit its ability to perform interventional queries. By using a simple extension to our analysis we permitted our active algorithm to choose to set any pair of nodes or any single node or no nodes at all. We compared this approach to random sampling and also uniformly choosing one of our possible queries (setting a single node, pair of nodes, or no nodes). Experiments were performed on the **Asia**, **Cancer**, and **Car** networks with an informed prior of 20 random observations. In this batch of experiments, we also experimented with different methods for choosing the candidate parents for a node X. As an alternative to using random nodes together with the true parents, we chose the $m = 5$ variables which had the highest individual mutual information with X. In practice this information could be obtained from observational data. Empirically, both methods of choosing the candidate parents gave very similar results, despite the fact that for one node in the **Car** network, a true parent of a node happened not to be chosen as a candidate parent with

[2]In general, information from observational data can easily be incorporated into our model simply by setting $\mathbf{Q}$ to be the empty set for each of the observational data instances. By Theorem 3.1, the update rule for these instances is equivalent to standard Bayesian updating of the model.

the mutual information method. We present the results using the mutual information criterion for choosing parents.

Figures 1(d), 1(e) and 1(g) show that in all networks our active method significantly outperforms the other methods. We also see, in Figures 1(f) and 1(h), that the prediction error graphs are very similar to the graphs of the edge entropy (Eq. (3)) based on our distribution over structures. This shows that the edge entropy is, indeed, a reasonable surrogate for predictive accuracy.

Figures 2(b), 2(c) and 2(d) show typical estimated causal edge probabilities in these experiments for random sampling, uniform querying and active querying respectively for the **Cancer** network (Fig. 2(a)). Figure 2(b) demonstrates that one requires more that just random observational data to learn the directions of many of the edges, and Fig. 2(d) shows that our active learning method creates better estimates of the causal interactions between variables than uniform querying. In fact, in some of the trials our active method recovered the edges and direction perfectly (when discarding low probablity edges) and was the only method that was able to do so given the limitation of just 50 queries. Also, our active method tends to be much better at not placing edges between variables that are only indirectly causally related; for instance in the network distribution learned by the active method (summarized in Fig. 2(d)), the probability of an edge from *Cancer* to *Coma* is only 3% and from *Cancer* to *Papilledema* only 4%.

6 Discussion and Conclusions

This paper introduces the problem of active learning of Bayesian network structure using interventional queries. We have presented a formal framework for this task, and a resulting algorithm for adaptively selecting which queries to perform. We have shown that our active method provides substantially better predictions regarding structure than both random sampling, and a process by which interventional queries are selected at random. Somewhat surprisingly, our algorithm achieves significant improvements over these other approaches even when it is restricted to querying roots in the network, and therefore cannot exploit the advantage of intervening in the model.

There is a significant body of work on the design of experiments in the field of optimal experimental design [Atkinson and Bailey, 2001]; there, the focus is not on learning the causal structure of a domain, and the experiment design is typically fixed in advanced, rather than selected actively. Active learning in Bayesian networks has been considered by Tong and Koller [2001] for parameter estimation where the structure is assumed to be known. There have been several studies on learning causal models from purely observational data [Spirtes *et al.*, 1993; Heckerman *et al.*, 1997]. Cooper and Yoo [1999] consider learning the structure of causal networks from a mixture of experimental and observational data, but in a non-active learning setting. Furthermore, although they derive a scoring metric for full networks, they only apply their technique to learn the relationship between single pairs of variables which they further assume are not confounded. On the other hand, our framework permits combining observational and experimental data for learning the structure over *all* variables in our domain, allowing us to distinguish the structure at a much finer level, taking into consideration both indirect causation and confounding influences.

There are many interesting directions in which we can extend our work, e.g., a treatment of continuous variables, temporal processes and different costs for different queries. Most interesting is the problem of dealing with hidden variables and missing data. We might use active learning to decide which extra variable to observe or which extra piece of missing data we should try to obtain in order to best learn the model. Another exciting direction is the potential of using active learning in order to try to uncover the existence of a hidden variable in our domain.

Acknowledgements The experiments were performed using the PHROG system, developed primarily by Lise Getoor, Uri Lerner, and Ben Taskar. The work was supported by DARPA's *Information Assurance* program under subcontract to SRI International, and by ONR MURI N00014-00-1-0637.

References

[Atkinson and Bailey, 2001] A. C. Atkinson and R. A. Bailey. One hundred years of the design of experiments on and off the pages of "Biometrika". *Biometrika*, 2001. In press.

[Buntine, 1991] W. Buntine. Theory refinement on Bayesian Networks. In *Proc. UAI*, 1991.

[Cooper and Yoo, 1999] G. F. Cooper and C. Yoo. Causal discovery from a mixture of experimental and observational data. In *Proc. UAI*, 1999.

[Friedman and Koller, 2000] N. Friedman and D. Koller. Being Bayesian about network structure. In *Proc. UAI*, 2000.

[Friedman *et al.*, 1999] N. Friedman, I. Nachman, and D. Pe'er. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Proc. UAI*, 1999.

[Heckerman *et al.*, 1995] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

[Heckerman *et al.*, 1997] D. Heckerman, C. Meek, and G. Cooper. A Bayesian approach to causal discovery. Technical Report MSR-TR-97-05, Microsoft Research, 1997.

[Lauritzen and Spiegelhalter, 1988] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statistical Society*, B 50(2), 1988.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[Pearl, 2000] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[Spirtes *et al.*, 1993] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. MIT Press, 1993.

[Tong and Koller, 2001] S. Tong and D. Koller. Active learning for parameter estimation in Bayesian networks. In *Proc. NIPS 13*, 2001. To appear.

Probabilistic Classification and Clustering in Relational Data

Ben Taskar
Computer Science Dept.
Stanford University
Stanford, CA 94305
btaskar@cs.stanford.edu

Eran Segal
Computer Science Dept.
Stanford University
Stanford, CA 94305
erans@cs.stanford.edu

Daphne Koller
Computer Science Dept.
Stanford University
Stanford, CA 94305
koller@cs.stanford.edu

Abstract

Supervised and unsupervised learning methods have traditionally focused on data consisting of *independent* instances of a single type. However, many real-world domains are best described by relational models in which instances of multiple types are *related* to each other in complex ways. For example, in a scientific paper domain, papers are related to each other via citation, and are also related to their authors. In this case, the label of one entity (e.g., the topic of the paper) is often correlated with the labels of related entities. We propose a general class of models for classification and clustering in relational domains that capture probabilistic dependencies between related instances. We show how to learn such models efficiently from data. We present empirical results on two real world data sets. Our experiments in a *transductive* classification setting indicate that accuracy can be significantly improved by modeling relational dependencies. Our algorithm automatically induces a very natural behavior, where our knowledge about one instance helps us classify related ones, which in turn help us classify others. In an unsupervised setting, our models produced coherent clusters with a very natural interpretation, even for instance types that do not have any attributes.

1 Introduction

Most supervised and unsupervised learning methods assume that data instances are independent and identically distributed (IID). Numerous classification and clustering approaches have been designed to work on such "flat" data, where each data instance is a fixed-length vector of attribute values (see [Duda *et al.*, 2000] for a survey). However, many real-world data sets are much richer in structure, involving instances of multiple types that are related to each other. Hypertext is one example, where web pages are connected by links. Another example is a domain of scientific papers, where papers are related to each other via citation, and are also related to their authors. The IID assumption is clearly violated for two papers written by the same author or two papers linked by citation, which are likely to have the same topic.

Recently, there has been a growing interest in learning techniques for more richly structured datasets. Relational links between instances provide a unique source of information that has been proved useful for both classification and clustering in the hypertext domain [Slattery and Craven, 1998; Kleinberg, 1998]. Intuitively, relational learning methods attempt to use our knowledge about one object to reach conclusions about other, related objects. For example, we would like to propagate information about the topic of a paper p to papers that it cites. These, in turn, would propagate information to papers that they cite. We would also like to use information about p's topic to help us reach conclusion about the research area of p's author, and about the topics of other papers written by that author.

Several authors have proposed relational classification methods along the lines of this "influence propagation" idea. Neville and Jensen [2000] present an *iterative classification* algorithm which essentially implements this process exactly, by iteratively assigning labels to test instances the classifier is confident about, and using these labels to classify related instances. Slattery and Mitchell [2000] propose an iterative algorithm called FOIL-HUBS for the problem of classifying web pages, e.g., as belonging to a university student or not. However, none of these approaches proposes a single coherent model of the correlations between different related instances. Hence they are forced to provide a purely procedural approach, where the results of different classification steps or algorithms are combined without a unifying principle.

In clustering, the emphasis so far has been on *dyadic* data, such as word-document co-occurrence [Hofmann and Puzicha, 1999], document citations [Cohn and Chang, 2000], web links [Cohn and Hofmann, 2001; Kleinberg, 1998], and gene expression data. Kleinberg's "Hubs and Authorities" algorithm exploits the link structure to define a mutually reinforcing relationship between hub and authority pages, where a good hub page points to many good authorities and a good authority page is pointed to by many good hubs.

These techniques can be viewed as relational clustering methods for one or two "types" of instances (e.g., web pages, documents and words), with a single relation between them (e.g., hyperlinks, word occurrence). However, we would like to model richer structures present in many real world domains with multiple types of instances and complex relationships between them. For example, in a movie database the instance types might be movies, actors, directors, and producers. Instances of the same type may also be directly related. In a scientific paper database, a paper is described by its set of words and its relations to the papers it cites (as well as to the authors who wrote it). We would like to identify, for each instance type, sub-populations (or segments) of instances that are similar in both their attributes and their relations to other

instances.

In this paper, we propose a general class of generative probabilistic models for classification and clustering in relational data. The key to our approach is the use of a single probabilistic model for the entire database that captures interactions between instances in the domain. Our work builds on the framework of *Probabilistic Relational Models (PRMs)* of Koller and Pfeffer [1998] that extend Bayesian networks to a relational setting. PRMs provide a language that allows us to capture probabilistic dependencies between related instances in a coherent way. In particular, we use it to allow dependencies between the class variables of related instances, providing a principled mechanism for propagating information between them.

Like all generative probabilistic models, our models accommodate the entire spectrum between purely supervised classification and purely unsupervised clustering. Thus, we can learn from data where some instances have a class label and other do not. We can also deal with cases where one (or more) of the instance types does not have an observed class attribute by introducing a new latent class variable to represent the (unobserved) cluster. Note that, in relational models, it is often impossible to segment the data into a training and test set that are *independent* of each other since the training and test instances may be interconnected. Using naive random sampling to select training instances is very likely to sever links between instances in the training and test set data. We circumvent this difficulty by using a transductive learning setting, where we use the test data, albeit without the labels, in the training phase. Hence, even if all the instance types have observed class attributes, the training phase involves learning with latent variables.

We provide an approximate EM algorithm for learning such PRMs with latent variables from a relational database. This task is quite complex: Our models induce a complex web of dependencies between the latent variables of all of the entities in the data, rendering standard approaches intractable. We provide an efficient approximate algorithm that scales linearly with the number of instances, and thus can be applied to large data sets.

We present experimental results for our approach on two domains: a dataset of scientific papers and authors and a database of movies, actors and directors. Our classification experiments show that the relational information provides a substantial boost in accuracy. Applied to a clustering task, we show that our methods are able to exploit the relational structure and find coherent clusters even for instance types that do not have any attributes.

2 Generative models for relational data

Probabilistic classification and clustering are often viewed from a generative perspective as a density estimation task. Data instances are assumed to be independent and identically distributed (IID) samples from a mixture model distribution. Each instance belongs to exactly one of k classes or clusters. In clustering, a latent class random variable is associated with the instance to indicate its cluster. Other attributes of an instance are then assumed to be samples from a distribution associated with its class. A simple yet powerful model often used for this distribution is the Naive Bayes model. In the Naive Bayes model, the attributes of each instance are assumed to be conditionally independent given the class variable. Although this independence assumption is often unrealistic, this model has nevertheless proven to be robust and effective for classification and clustering across a wide range of applications [Duda *et al.*, 2000; Cheeseman and Stutz, 1995]. Both classification and clustering involve estimation of the parameters of the Naive Bayes model; however, clustering is significantly more difficult due to the presence of latent variables.

The IID assumption made by these standard classification and clustering models is inappropriate in rich relational domains, where different instances are related to each other, and are therefore likely to be correlated. In this section, we describe a probabilistic model for classification and clustering in relational domains, where entities are related to each other. Our construction utilizes the framework of *probabilistic relational models (PRMs)* [Koller and Pfeffer, 1998; Friedman *et al.*, 1999].

2.1 Probabilistic Relational Models

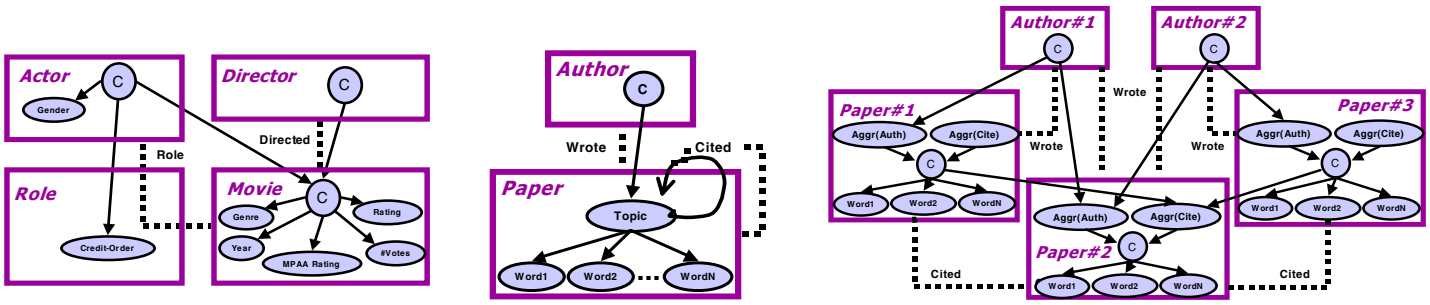
A PRM is a template for a probability distribution over a relational database of a given schema. It specifies probabilistic models for different classes of entities, including probabilistic dependencies between related objects. Given a set of instances and relations between them, a PRM defines a joint probability distribution over the attributes of the instances.

Relational Schema. A relational schema describes attributes and relations of a set of instance types $\mathcal{X} = \{X_1, \ldots, X_n\}$. Each type X is associated with a set of *attributes* $\mathcal{A}(X)$. Each type X is also associated with a set $\mathcal{R}(X)$ of typed binary relations $R(X, Y)$. We associate each relation R with the type X of its first argument, allowing us to use the relation as a set-valued function, whose value $x.R$ is the set of instances $y \in Y$ related to an instance x. For example, for an actor a, $a.Role$ is the set of movies in which the actor has appeared.

In certain cases, relations might have attributes of their own. For example, the "Role" relation might be associated with the attribute *Credit-Order*, which indicates the ranking of the actor in the credits. We can introduce an explicit type corresponding to the relation. In this case, a relation object is itself related to both of its arguments. For example, if one of the role objects is "Meryl Streep in Sophie's Choice", this role object would be related to the actor object "Meryl Streep" and the movie object "Sophie's Choice". By definition, these relations are many-to-one. It will be useful to distinguish between *entity* types (such as Person or Movie), and *relation* types (such as Role).

An *instantiation* $\mathcal{I}$ specifies the set of objects in each type, the relations that hold between them, and the values of the attributes for all of the objects. A *skeleton* σ specifies only the objects and the relations. We will use $\sigma(X)$ to denote the set of objects of type X.

Probabilistic Model. A probabilistic relational model Π specifies a probability distribution over a set of instantiations $\mathcal{I}$ of the relational schema. More precisely, a PRM is a tem-



(a) (b) (c)

Figure 1: (a) Model for IMDB domain; (b) Model for Cora domain; (c) Fragment of unrolled network for Cora model.

plate, which is instantiated for different skeletons σ. The result of this instantiation is a probabilistic model over a set of random variables corresponding to all of the attributes of all of the objects in the skeleton. We can view a PRM as a compact way of representing a Bayesian network for any skeleton over this schema.

A PRM consists of a qualitative dependency structure, $\mathcal{S}$, and the parameters associated with it, $\boldsymbol{\theta}_{\mathcal{S}}$. The dependency structure is defined by associating with each attribute $X.A$ a set of *parents* $\mathrm{Pa}(X.A)$. Each parent of $X.A$ has the form of either $X.B$ or $X.R.B$ for $R \in \mathcal{R}(X)$. (PRMs also allow dependencies along chains of relations, but we have chosen to omit those for simplicity of presentation.)

For a given skeleton σ, the PRM structure induces an *unrolled* Bayesian network over the random variables $x.A$. For every object $x \in \sigma(X)$, $x.A$ depends probabilistically on parents of the form $x.B$ or $x.R.B$. Note that if R is not single-valued, then $x.R.B$ is actually a set of random variables, one for each $y \in x.R$. We address this problem by interpreting the dependence of $X.A$ on $X.R.B$ as dependence on an aggregate function (e.g., mode or mean) of the multiset of values of these variables (see below).

The quantitative part of the PRM specifies the parameterization of the model. Given a set of parents for an attribute, we can define a local probability model by associating with it a *conditional probability distribution (CPD)*. For each attribute we have a CPD that specifies $P(X.A \mid \mathrm{Pa}(X.A))$. When one of the parents is of the form $X.R.B$ and R is many-valued, the CPD represents the dependence on the value of the aggregate. The CPD for $X.A$ is used for $x.A$ in the unrolled network, for every x in X. Thus, the CPD for $X.A$ is repeated many times in the network.

Aggregates. There are many possible choices of aggregation operator to allow dependencies on a set of variables. An obvious choice for categorical variables is the mode aggregate, which computes the most common value of its parents. More precisely, consider some variable Y whose parents $\mathbf{X} = \{X_1 \ldots X_n\}$ we wish to aggregate into a single variable A. Let the domain of each X_i be $\{v_1, \ldots, v_k\}$, and note that A has the same domain. The effect of the mode aggregator is as follows: We define a distribution $P(Y \mid v_i)$ for each i; given a multiset of values for $X_1, \ldots, X_n$, we use the distribution $P(Y \mid v_i)$ for the value v_i which is the most common in this multiset.

The mode aggregator is not very sensitive to the distribution of values of its parents; for example, it cannot differentiate between a highly skewed and a fairly uniform set of values that have the same most frequent value. An aggregate that better reflects the value distribution is a *stochastic mode* aggregator. In this case, we still define a set of distributions $P(Y \mid v_i)$, but the effect of the aggregator is that $P(Y \mid X_1, \ldots, X_n)$ is a *weighted average* of these distributions, where the weight of v_i is the frequency of this value within $X_1, \ldots, X_n$. We accomplish this behavior by using an aggregate variable $A = \textit{Stochastic-Mode}(X_1, \ldots, X_n)$, defined as follows. The aggregate variable also takes on values in $\{v_1, \ldots, v_k\}$. Let $t_j(\mathbf{X})$ be the number of variables X_i that take on the value v_j. Then we define $P(A = v_j \mid \mathbf{X}) = t_j(\mathbf{X})/n$. It is easy to verify that this aggregator has exactly the desired effect.

We note that this aggregate can also be viewed as a randomized selector node that chooses one of its parents uniformly at random and takes on its value. One appealing consequence is that, like min or max, the stochastic model can be decomposed to allow its representation as a CPD to scale linearly with the number of parents. We simply decompose the aggregate in a cascading binary tree. The first layer computes aggregates of disjoint pairs, with each aggregate randomly selecting the value of one of its parents; the following layer repeats the procedure for disjoint pairs of results from the first layer, and so on. (This construction can also be extended to cases where the number of variables is not a power of 2; we omit details for lack of space.)

2.2 Classification and clustering models

We use the PRM framework as the basis of our models for relational classification and clustering. As in the "flat" probabilistic generative approaches, our approach is based on the use of a special variable to represent the class or the cluster. This variable is the standard "class" variable in the classification task. As usual, we deal with the clustering task by introducing a new latent class variable. Thus, for each entity class X we have a designated attribute $X.C$ in $\mathcal{A}(X)$.

As in flat classification and clustering, we define the attributes of X to depend on the class variable. For simplicity, we choose the Naive Bayes dependency model for the other attributes: For each attribute $X.A$, the only parent of $X.A$

is $X.C$. Note that we have only defined class attributes for entity types. We connect the attributes of relation types to the class attributes of the two associated entity types. Thus, for example, an attribute such as *Credit-Order* in the relation class Role will depend on the class attributes of Actor and Movie. Note that, as the dependence in this case is single-valued by definition, no aggregates are necessary. Most interestingly, we also allow direct dependence between class attributes of related entities. Thus, for example, we could allow a dependence of Movie.C on Actor.C, or vice versa. In this case, as the relation is many-to-many, we use aggregates, as described above.

Fig. 1(a) shows a simple model for a movie dataset, extracted from the *Internet Movie Database (IMDB)* (`www.imdb.com`). We see that "Role" is both a class on its own, as well as defining the relation between movies and actors. We have chosen, in this case, not to have the attribute Role.*Credit-Order* depend on the class of movies, but only of actors. Fig. 1(b) shows a model for a domain of scientific papers and authors, derived from the Cora dataset [McCallum *et al.*, 2000] (`cora.whizbang.com`). In this case, we see that the "Cites" relation connects two objects of the same type. We have chosen to make the class attribute of the cited paper depend on the class attribute of the citing paper. Note that this dependency appears cyclic at the type level. However, recall that this model is only a template, which is instantiated for particular skeletons to produce an unrolled network; Fig. 1(c) shows a fragment of such a network. If we do not have "citation cycles" in the domain, then this unrolled network is acyclic, and the PRM induces a coherent probability model over the random variables of the skeleton. (See [Friedman *et al.*, 1999] for more details.)

We can also use latent variable models to represent dyadic clustering. Consider, for example, a domain where we have people and movies, and a relation between them that corresponds to a person rating a movie. In this case, we will have a class Vote, corresponding to the relation, with the attribute *Rating* representing the actual rating given. This attribute will depend on the cluster attributes of both Movie and Person, leading naturally to a two-sided clustering model. However, our approach is flexible enough to accommodate a much richer model, e.g., where we also have other attributes of person, and perhaps an entire relational model for movies, such as shown in Fig. 1(a). Our approach will take all of this information into consideration when constructing the clusters.

3 Learning the models

We now show how we learn our models from data. Our training set D consists of a partial instantiation of the schema, one where everything except the values of some or all the class attributes is given. We can view this data as a single large "mega-instance" of the model, with a large number of missing values. Note that we cannot view the data as a set of independent instances corresponding to the objects in the model. In our setting, we typically assume that the structure of our latent variable model is given, as described in Section 2.2. Thus, our task is parameter estimation.

3.1 Parameter estimation

In this case, we assume that we are given the probabilistic dependency structure $\mathcal{S}$, and need only estimate the parameters $\boldsymbol{\theta}_{\mathcal{S}}$, i.e., the CPDs of the attributes. A standard approach is to use *maximum likelihood* (ML) estimation, i.e., to find $\boldsymbol{\theta}_{\mathcal{S}}$ that maximize $P(D \mid \boldsymbol{\theta}_{\mathcal{S}})$.

If we had a complete instantiation $\mathcal{I}$, the likelihood function has a unique global maximum. The maximum likelihood parameters can be found very easily simply by counting occurrences in the data. Recall that all of the objects in the same class share the same CPD. Thus, to estimate the parameter for $P(X.A \mid \mathrm{Pa}(X.A))$, we simply consider all objects x of class X, and count the number of times that each combination $v, \mathbf{u}$ that $x.A$ and its parents jointly take. These counts are known as *sufficient statistics*. See [Friedman *et al.*, 1999] for details.

The case of incomplete data is substantially more complex. In this case, the likelihood function has multiple local maxima, and no general method exists for finding the global maximum. The *Expectation Maximization (EM)* algorithm [Dempster *et al.*, 1977], provides an approach for finding a local maximum of the likelihood function. Starting from an initial guess $\boldsymbol{\theta}^{(0)}$ for the parameters, EM iterates the following two steps. The E-step computes the distribution over the unobserved variables given the observed data and the current estimate of the parameters. Letting $\mathbf{C}$ be the set of unobserved cluster variables, we compute $P(\mathbf{C} \mid D, \boldsymbol{\theta}^{(t-1)})$, from which it can compute the *expected* sufficient statistics:

$$N_{X.A}[v, \mathbf{u}] = \sum_{x \in \sigma(X)} P(x.A = v \mid \mathrm{Pa}(x.A) = \mathbf{u}, D, \boldsymbol{\theta}^{(t-1)})$$

To compute the posterior distribution over the hidden variables, we must run inference over the model. The M-step reestimates the parameters by maximizing the likelihood with respect to the distribution computed in the E-step.

$$\theta^{X.A}_{v|\mathbf{u}} = \frac{N_{X.A}[v, \mathbf{u}]}{\sum_v N_{X.A}[v, \mathbf{u}]}$$

3.2 Belief Propagation for E step

To perform the E step, we need to compute the posterior distribution over the unobserved variables given our data. This inference is over the unrolled network defined in Section 2.2. We cannot decompose this task into separate inference tasks over the objects in the model, as they are all correlated. (In some cases, the unrolled network may have several connected components that can be treated separately; however, it will generally contain one or more large connected components.)

In general, the unrolled network can be fairly complex, involving many objects that are related in various ways. (In our experiments, the networks involve tens of thousands of nodes.) Exact inference over these networks is clearly impractical, so we must resort to approximate inference. There is a wide variety of approximation schemes for Bayesian networks. For various reasons (some of which are described below), we chose to use *belief propagation*. Belief Propagation (BP) is a local message passing algorithm introduced by Pearl [Pearl, 1988]. It is guaranteed to converge to the correct marginal probabilities for each node only for singly connected Bayesian networks. However, empirical results [Murphy and Weiss, 1999] show that it often converges in general

networks, and when it does, the marginals are a good approximation to the correct posteriors. (When BP does not converge, the marginals for some nodes can be very inaccurate. This happens very rarely in our experiments and does not affect convergence of EM.)

We provide a brief outline of one variant of BP, referring to [Murphy and Weiss, 1999] for more details. Consider a Bayesian network over some set of nodes (which in our case would be the variables $x.A$). We first convert the graph into a *family graph*, with a node F_i for each variable X_i in the BN, containing X_i and its parents. Two nodes are connected if they have some variable in common. The CPD of X_i is associated with F_i. Let ϕ_i represent the factor defined by the CPD; i.e., if F_i contains the variables $X, Y_1, \ldots, Y_k$, then ϕ_i is a function from the domains of these variables to $[0, 1]$. We also define ψ_i to be a factor over X_i that encompasses our evidence about X_i: $\psi_i(X_i) \equiv 1$ if X_i is not observed. If we observe $X_i = x$, we have that $\psi_i(x) = 1$ and 0 elsewhere. Our posterior distribution is then $\alpha \prod_i \phi_i \times \prod_i \psi_i$, where α is a normalizing constant.

The belief propagation algorithm is now very simple. At each iteration, all the family nodes simultaneously send message to all others, as follows:

$$m_{ij}(F_i \cap F_j) \leftarrow \alpha \sum_{F_i - F_j} \phi_i \psi_i \prod_{k \in N(i) - \{j\}} m_{ki}$$

where α is a (different) normalizing constant and $N(i)$ is the set of families that are neighbors of F_i in the family graph. At any point in the algorithm, our marginal distribution about any family F_i is $b_i = \alpha \phi_i \psi_i \prod_{k \in N(i)} m_{ki}$. This process is repeated until the beliefs converge.

After convergence, the b_i give us the marginal distribution over each of the families in the unrolled network. These marginals are precisely what we need for the computation of the expected sufficient statistics.

We note that occasionally BP does not converge; to alleviate this problem, we start the EM algorithm from several different starting points (initial guesses). As our results in Section 5 show, this approach works well in practice.

4 Influence propagation over relations

Among the strong motivations for using a relational model is its ability to model dependencies between related instances. As described in the introduction, we would like to propagate information about one object to help us reach conclusions about other, related objects. Recently, several papers have proposed a process along the lines of this "influence propagation" idea. Neville and Jensen [2000] propose an *iterative classification* algorithm which builds a classifier based on a fully observed relational training set; the classifier uses both base attributes and more relational attributes (e.g., the number of related entities of a given type). It then uses this classifier on a test set where the base attributes are observed, but the class variables are not. Those instances that are classified with high confidence are temporarily labeled with the predicted class; the classification algorithm is then rerun, with the additional information. The process repeats several times. The classification accuracy is shown to improve substantially as the process iterates.

Slattery and Mitchell [2000] propose an application of this idea to the problem of classifying web pages, e.g., as belonging to a university student or not. They first train a classifier on a set of labeled documents, and use it to classify documents in the test set. To classify more documents in the test set, they suggest combining the classification of the test set pages and the relational structure of the test set. As a motivating example, they describe a scenario where there exists a page that points to several other pages, some of which were classified as student home pages. Their approach tries to identify this page as a student directory page, and conclude that other pages to which it points are also more likely to be student pages. They show that classification accuracy improves by exploiting the relational structure.

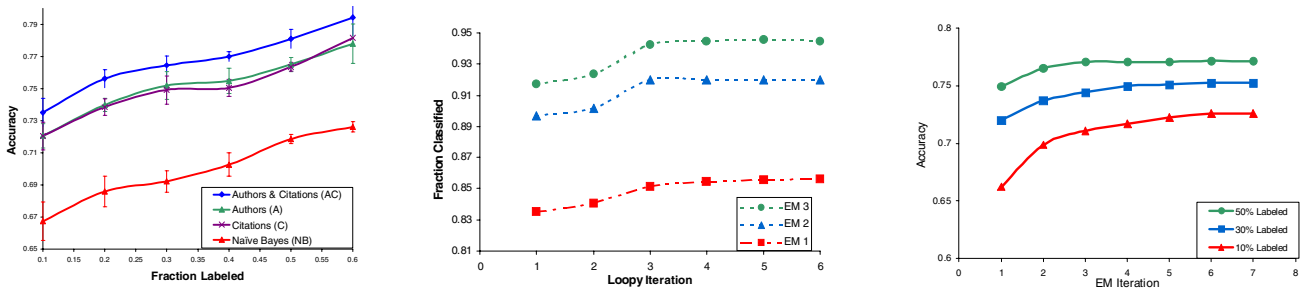
Neither of these approaches proposes a single coherent model of the dependencies between related objects and thus combine different classification steps or algorithms without a unifying principle. Our approach achieves the influence propagation effect through the probabilistic influences induced by the unrolled Bayesian network over the instances in our domain. For example, in the Cora domain, our network models correlations between the topics of papers that cite each other. Thus, our beliefs about the topic of one paper will influence our beliefs about the topic of its related papers. In general, probabilistic influence "flows" through active paths in the unrolled network, allowing beliefs about one cluster to influence others to which it is related (directly or indirectly). Moreover, the use of belief propagation implements this effect directly. By propagating a local message from one family to another in the family graph network, the algorithm propagates our beliefs about one variable to other variables to which it is directly connected. We demonstrate this property in the next section.

This spreading influence is particularly useful in our framework due to the application of the EM algorithm. The EM algorithm constructs a sequence of models, using the probabilities derived from the belief propagation algorithm to train a new model. Hence, we not only use our probabilistic inference process to spread the information in the relational structure, we then use the results to construct a better classifier, which in turn allows us to obtain even better results, etc. From a different perspective, we are using the structure in the test set not only to provide better classifications, but also to learn a better classifier. As we show below, this process results in substantial improvements in accuracy over the iterations of EM. We note that this bootstrapping ability arises very naturally in the probabilistic framework, where it is also associated with compelling convergence guarantees.

5 Experiments

We evaluated our method on the Cora and IMDB data sets.

Cora. The structure of the Cora dataset, and the model we used, are shown in Fig. 1(b,c). For our experiments, we selected a subset of 4187 papers from the Machine Learning category, along with 1454 of their authors. These papers are classified into seven topics: Probablistic Methods, Neural networks, Reinforcement Learning, Rule Learning, Case-Based, and Theory.



(a) (b) (c)

Figure 2: (a) Comparison of classification accuracies; (b) Influence propagation in BP; (c) Accuracy improvement in EM.

We evaluated the ability of our algorithm to use the relational structure to aid in classification. We took our entire data set, and hid the classifications for all but a fraction of the papers. We then constructed our model based on all of this data, including the documents whose topics were unobserved. The resulting model was used to classify the topic for the test documents. In effect, we are performing a type of *transduction*, where the test set is also used to train the model (albeit without the class labels).

To investigate how our method benefits from exploiting the relational structure, we considered four different models which vary in the amount of relational information they use. The baseline model does not use relational information at all. It is a standard multinomial Naive Bayes model (**NB**) over the set of words (bag of words model) in the abstract. The full model (**AC**) was shown in Fig. 1(b); it makes use of both the authors and citations. The other two models are fragments of **AC**: model **A** incorporates only the author information (eliminating the citation relation from the model), and model **C** only citations. All four models were trained using EM; model **NB** was trained using exact EM and the others using our algorithm of Section 3. We initialized the CPDs for the word attributes using the CPDs in a Naive Bayes model that was trained only on the observed portion of the data set. All models were initialized with the same CPDs.

We varied the percentage of labeled papers, ranging from 10% to 60%. For each different percentage, we tested the classification accuracy over five random training/test splits. The results are shown in Fig. 2(a). Each point is the average of the accuracy on the five runs, and the error bars correspond to the standard error. As can be seen, incorporating more relational dependencies significantly improves classification accuracy. Both **A** and **C** outperform the baseline model, and the combined model **AC** achieves by far the highest accuracy.

As discussed in Section 4, the local message passing of loopy belief propagation (BP) resembles the process of "spreading" the influence of beliefs for a particular instance to its related instances. For example, suppose paper x cites several labeled papers. Upon initialization, we have some initial belief about the topic of x from its words alone. However, after the first iteration, this belief will be updated to reflect the labels of the papers it cites, and is likely to become more peaked around a single value, increasing the confidence in x's topic. In the following iteration, unlabeled papers that cite x (as well as unlabeled papers that x cites) will be updated to reflect the increased confidence about the topic of x, and so on. To measure this effect, we examine the belief state of the topic variable of the unlabeled papers after every iteration of loopy belief propagation. For every iteration, we report the fraction of variables whose topic can be determined with high confidence, i.e., whose belief for a single topic is above a threshold of 0.9. Fig. 2(b) shows several series of these measurements on a dataset with 10% labeled papers. The series show BP iterations performed within the first, third and seventh iteration of EM. Each series shows a gradual increase of the fraction of papers whose topics we are confident in. The accuracy on those high-confidence papers is fairly constant over the iterations — around 0.7, 0.735, and 0.74 for the first, third and seventh iteration of EM, respectively.

Loopy belief propagation is an approximation to the inference required for the E step of EM. Although loopy BP is not guaranteed to converge, in our experiments, it generally converges to a solution which is good enough to allow EM to make progress. Indeed, Fig. 2(c) shows that the classification accuracy improves for every EM iteration. This figure also demonstrates the performance improvement obtained from bootstrapping the results of iterative classification, as discussed in Section 4.

IMDB. The attributes and relations in the IMDB database, and the latent variable model we used, are shown in are shown in Fig. 1(a); the *Genre* attribute actually refers to a set of 18 binary attributes (action, comedy, ...). Note that actors and directors have almost no descriptive attributes and hence cannot be clustered meaningfully without considering their relations. We selected a subset of this database that contains 1138 movies, 2446 actors, and 734 directors. In Fig. 5, we show two example clusters for each class, listing several highest confidence members of the clusters.

In general, clusters for movies consist of movies of predominantly of a particular genre, time period and popularity. For example, the first movie cluster shown can be labeled as classic musicals and children's films. The second cluster corresponds roughly to action/adventure/sci-fi movies. In our model, the clusters for actors and directors are relational in nature, since they are induced by the movie attributes. For example, the first cluster of actors consists primarily of action

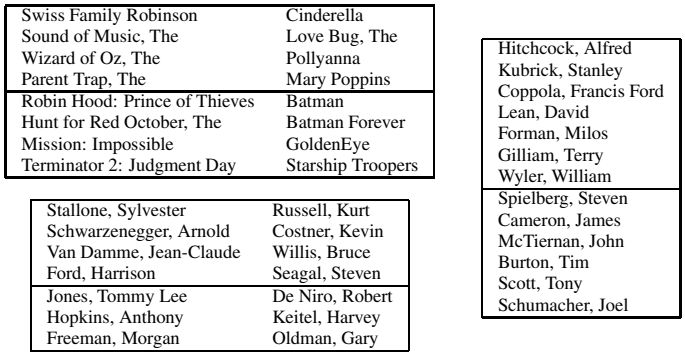


Figure 3: Clusters of Movies, Actors, and Directors

movie actors and the second of actors who primarily play in dramas. Similarly for directors, the first cluster corresponds to directors of dramas and while the second to directors of popular action and adventure films.

6 Discussion and Conclusions

Many real-world domains have a rich relational structure, with complex webs of interacting entities: the web; papers and authors; biomedical datasets; and more. Traditional machine learning algorithms ignore this rich relational structure, flattening it into a set of IID attribute vectors. Recently, however, there has been growing interest in learning methods that exploit the relational structure of the domain.

In this paper, we provide a general method for classification and clustering in richly structured data with instances and relations. Our approach has coherent probabilistic semantics, allowing us to build on powerful tools for probabilistic reasoning and learning. Our algorithm uses an effective combination of these techniques to provide linear scaling in the number of instances; it can thus be applied to very large domains.

We have shown in a transduction task that the relational information allows us to achieve substantially better accuracies than a standard "flat" classification scheme. We have also shown anecdotally that our algorithm constructs interesting clusters based on relational information. Finally, our approach induces a compelling behavior unique to relational settings: Because instances are *not* independent, information about some instances can be used to reach conclusions about others. Our approach is the first to provide a formal framework for this behavior.

There are many interesting extensions to this work. Most obvious is the problem of *model selection*. In this paper, we have used predetermined models, both the edges in the dependency model and the number of clusters of each latent class attribute. Our framework allows us to incorporate into our models a great deal of prior knowledge about the semantics of the underlying domains. However, when domain expertise is lacking, automatic model construction becomes crucial. We can extend our approach using techniques for model selection in Bayesian networks [Friedman, 1998; Cheeseman and Stutz, 1995], allowing our learning algorithm to select the model structure that best suits the data.

Acknowledgments. This work was supported by ONR contract N66001-97-C-8554 under DARPA's HPKB program. Eran Segal was also supported by a Stanford Graduate Fellowship (SGF).

References

[Cheeseman and Stutz, 1995] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. In Fayyad U., Piatesky-Shapiro G., Smyth P., and Uthuruasmy R., editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press, Menlo Park, CA, 1995.

[Cohn and Chang, 2000] D. Cohn and H. Chang. Probabilistically identifying authoritative documents. In *Proc. SIGIR*, 2000.

[Cohn and Hofmann, 2001] D. Cohn and T. Hofmann. The missing link: A probabilistic model of document content and hypertext connectivity. In *Proc. NIPS*, 2001. To appear.

[Dempster *et al.*, 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B 39:1–39, 1977.

[Duda *et al.*, 2000] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2000.

[Friedman *et al.*, 1999] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proc. IJCAI*, 1999.

[Friedman, 1998] N. Friedman. The Bayesian structural EM algorithm. In *Proc. UAI*, 1998.

[Hofmann and Puzicha, 1999] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proc. IJCAI*, 1999.

[Kleinberg, 1998] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[Koller and Pfeffer, 1998] D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proc. AAAI*, 1998.

[McCallum *et al.*, 2000] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. *Automating the construction of internet portals with machine learning*. Information Retrieval Journal, 3:127, 2000.

[Murphy and Weiss, 1999] K. Murphy and Y. Weiss. Loopy belief propagation for approximate inference: an empirical study. In *UAI*, 1999.

[Neville and Jensen, 2000] J. Neville and D. Jensen. Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20. AAAI Press, 2000.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[Slattery and Craven, 1998] S. Slattery and M. Craven. Combining statistical and relational methods in hypertext domains. In *Proc. ILP*, 1998.

[Slattery and Mitchell, 2000] S. Slattery and T. Mitchell. Discovering test set regularities in relational domains. In *Proc. ICML*, 2000.

MACHINE LEARNING AND DATA MINING

MACHINE LEARNING AND DATA MINING

Adaptive Web Navigation for Wireless Devices

Corin R. Anderson, Pedro Domingos, Daniel S. Weld
University of Washington, Seattle, WA, USA
{corin, pedrod, weld}@cs.washington.edu

Abstract

Visitors who browse the web from wireless PDAs, cell phones, and pagers are frequently stymied by web interfaces optimized for desktop PCs. Simply replacing graphics with text and reformatting tables does not solve the problem, because deep link structures can still require minutes to traverse.

In this paper we develop an algorithm, MINPATH, that automatically improves wireless web navigation by suggesting useful *shortcut links* in real time. MINPATH finds shortcuts by using a learned model of web visitor behavior to estimate the savings of shortcut links, and suggests only the few best links. We explore a variety of predictive models, including Naïve Bayes mixture models and mixtures of Markov models, and report empirical evidence that MINPATH finds useful shortcuts that save substantial navigational effort.

1 Introduction

Perkowitz and Etzioni [1997] challenged the AI community to build *adaptive web sites*: sites that automatically improve their organization and presentation by learning from visitor access patterns. In the spirit of this challenge, many research projects have been proposed and implemented [Perkowitz and Etzioni, 2000; Fink *et al.*, 1996; Yan *et al.*, 1996; Jühne *et al.*, 1998; Joachims *et al.*, 1997; Pazzani and Billsus, 1999; Sarukkai, 2000]. Many of these projects, like much of the Web today, assume the visitor is browsing with a large color display and fast network connection. In addition, these works typically assume that a visitor's interest in a site lies in viewing many pages of content, as opposed to a specific destination. Consequently, the adaptations they emphasize include graphical highlighting of existing hypertext links and automatic generation of indices that link to many pages.

However, a growing community of web visitors does not fit this mold: *wireless web visitors*, who browse content from cell phones, pagers, and wireless PDAs. These mobile devices do not have the same display or bandwidth capabilities as their desktop counterparts, but sites (adaptive or otherwise) nonetheless deliver the same content to both desktop PCs and mobile devices. Moreover, our experience with mobile visitors indicates that they seldom browse through many pages, but rather are interested only in specific destinations, and would like to reach those pages by following as few links as possible. While this observation is also largely true of desktop visitors, reducing the number of links followed is particularly poignant for mobile visitors for two reasons. First, simply finding a specific link on a page requires the visitor to spend considerable time scrolling through the page on a tiny screen. Second, because of the low bandwidth and high latency of wireless networks, following a link to even a simple page of text can take as long as tens of seconds. Consequently, navigating through even four or five different pages can easily take several minutes, frustrating visitors to the point of abandoning their effort.

We believe adaptive web sites hold a great promise for improving the web experience for wireless devices, and, conversely, that wireless web browsing is a "killer app" of adaptive web sites. In ongoing research [Anderson *et al.*, 2001], we have developed a general framework for adapting web sites for wireless visitors that takes into account the value visitors receive for viewing a page and the effort required to reach that page (*i.e.*, the amount of scrolling and number of links followed). In this paper, we explore a particular adaptation for use in our framework: automatically adding *shortcut links* to each page a visitor requests. We assume that wireless visitor behavior is dominated by *information gathering tasks*, which can be accomplished by viewing specific destination pages on the site. Shortcut links connect two pages not otherwise linked together and can help visitors reach these destinations as quickly as possible. For example, if $A \rightarrow B$ denotes a link, and the only path from A to D is $A \rightarrow B \rightarrow C \rightarrow D$, then a shortcut $A \rightarrow D$ saves the visitor over 66% of the navigation effort, and perhaps more if one counts the scrolling avoided on the interim pages. Of course, there is a tradeoff between the number of shortcut links and their usefulness. In an absurd example, if one added shortcuts between every pair of pages on the site, a visitor could reach any destination in one link, but finding the right link would be practically impossible. Thus, we concentrate on generating only a small number of high quality shortcuts, for example, only as many as will fit on the wireless device without scrolling.

This paper makes the following contributions:

- We present an algorithm, MINPATH, for automatically finding high-quality shortcuts for mobile visitors in real time. Offline, MINPATH learns a model of web usage based on server access logs, and uses this model at runtime to predict the visitor's ultimate destination. MIN-

PATH is based on the notion of the *expected savings* of a shortcut, which incorporates the probability that the link is useful and the amount of visitor effort saved.

- We evaluate a variety of visitor models, including Naïve Bayes mixture models and mixtures of Markov models, and discuss their applicability in MINPATH.
- We provide experimental evidence that suggests a mixture of Markov models is the best model for MINPATH, and that MINPATH substantially reduces the number of links mobile visitors need to follow, and thus their navigational effort.

In the next section, we define our problem and present the MINPATH algorithm for finding shortcuts. Section 3 explores variations on the models for predicting web usage and section 4 evaluates MINPATH using these models. Section 5 discusses related research, and we conclude in section 6.

2 Finding shortcuts with MinPath

We begin by defining terminology, to facilitate the discussion.

2.1 Definitions

A *trail* [Wexelblat and Maes, 1999] is a sequence of page requests made by a single visitor that is coherent in time and space. Coherence in time requires that each subsequent request in the trail occurs within some fixed time window of the previous request. Coherence in space requires that each subsequent request be the destination of some link on the previous page. More precisely, if we denote the time of the request for page p_i as $\mathsf{time}(p_i)$, then a trail $T = \langle p_0, p_1, \ldots p_n \rangle$ is a sequence of page requests for which:

- $\forall\, i, 0 \leq i < n, p_i \rightarrow p_{i+1}$ exists; and
- $\forall\, i, 0 \leq i < n,$ $\mathsf{time}(p_i) \leq \mathsf{time}(p_{i+1}) \leq \mathsf{time}(p_i) + \textit{timeout}$

The *length* of a trail is n, the number of links followed. From the perspective of the adaptive web site which is watching a visitor's behavior midway through the trail, only a *prefix*, $\langle p_0, \ldots, p_i \rangle$ is known. The trail *suffix*, $\langle p_{i+1}, \ldots, p_n \rangle$, needs to be hypothesized by the adaptive web site.

2.2 Finding shortcuts

The objective of our work is to provide shortcut links to visitors to help shorten long trails. Our system adds shortcut links to every page the visitor requests. Ideally, the shortcuts suggested will help the visitor reach the destination of the trail with as few links as possible. We state the shortcut link selection problem precisely as:

- **Given:** a visitor V, a trail prefix $\langle p_0, \ldots, p_i \rangle$, and a maximum number of shortcuts m;
- **Output:** a list of shortcuts $p_i \rightarrow q_1, \ldots, p_i \rightarrow q_m$ that *minimizes* the number of links the visitor must follow between p_i and the visitor's destination.

The last page in the trail prefix, p_i, is the page the visitor has requested most recently, and the page on which the shortcuts are placed. We calculate the *savings* that a single shortcut $p_i \rightarrow q$ offers as the number of links the visitor can avoid by following that shortcut. If we know the entire trail $T = \langle p_0, \ldots, p_i, \ldots, p_n \rangle$, then the number of links saved is:

$$\begin{cases} j - i - 1 & \text{if } q = p_j \text{ for some } i < j \leq n \\ 0 & \text{otherwise} \end{cases}$$

That is, if the shortcut leads to a page further along the trail, then the savings is the number of links skipped (we subtract one because the visitor must still follow a link — the shortcut link). If the shortcut leads elsewhere, then it offers no savings.

2.3 The MinPath algorithm

If one had knowledge of the complete trail $\langle p_0, \ldots, p_i, \ldots p_n \rangle$, selecting the best shortcut at any page p_i would be easy: simply, $p_i \rightarrow p_n$. Of course, at runtime, a visitor has viewed only a trail prefix, and the adaptive web site must infer the remaining pages. Our approach relies on a model of the visitor's behavior to compute a probability for every possible trail suffix $\langle q_{i+1}, \ldots, q_n \rangle$ on the site. Intuitively, these suffixes are all the possible trails originating from p_i. Given a suffix and its probability, we assign an *expected savings* to the shortcut $p_i \rightarrow q_j$ to each q_j in the suffix as the product of the probability of the suffix and the number of links saved by the shortcut. Note that a particular shortcut $p_i \rightarrow q_j$ may appear in many trail suffixes (*i.e.*, many trail suffixes may pass through the same page q_j), and so the expected savings of a shortcut is the sum of the savings of the shortcut for all suffixes.

A brief example will elucidate these ideas. Suppose that a visitor has requested the trail prefix $\langle A, B, C \rangle$ and we wish to find shortcuts to add to page C. Suppose that our model of the visitor indicates there are exactly two sequences of pages the visitor may complete the trail with: $\langle D, E, F, G, H \rangle$, with a probability of 0.6, and $\langle I, J, H, K \rangle$ with a probability of 0.4. The expected savings from the shortcut $C \rightarrow E$ would be $0.6 \times 1 = 0.6$, because the trail with page E occurs with probability 0.6 and the shortcut saves only one link. The expected savings for shortcut $C \rightarrow H$ includes a contribution from both suffixes: $0.6 \times 4 + 0.4 \times 2 = 2.4 + 0.8 = 3.2$.

The MINPATH algorithm is shown in Table 1. The ExpectedSavings function constructs the trail suffixes by traversing the directed graph induced by the web site's link structure (often called the "web graph"). Starting at the page last requested by the visitor, p_i, ExpectedSavings computes the probability of following each link and recursively traverses the graph until the probability of viewing a page falls below a threshold, or a depth bound is exceeded. The savings at each page (*CurrentSavings*) is the product of the probability, P_s, of reaching that page along suffix T_s and the number of links saved, $l - 1$. The MinPath function collates the results and returns the best m shortcuts. The next section describes how we obtain the model required by MINPATH.

3 Predictive Models

The key element to MINPATH's success is the predictive model of web usage; in this section, we describe the models we have evaluated. The probabilistic model MINPATH uses must predict the next web page request p_i given a trail prefix $\langle p_0, \ldots, p_{i-1} \rangle$ and the visitor's identity V (the identity can lead to information about past behavior at the site, demographics, etc.): $P(p_i = q | \langle p_0, \ldots, p_{i-1} \rangle, V)$. Of course, a model may condition this probability on only part or even none of the available data; we explore these and other variations in this section. To simplify our discussion, we define a "sink" page that visitors (implicitly) request when they

Inputs:
T *Observed trail prefix* $\langle p_0, \ldots, p_i \rangle$
p_i *Most recent page requested*
V *Visitor identity*
m *Number of shortcuts to return*

MinPath(T, p_i, V, m)
  $S \leftarrow$ ExpectedSavings($p_i, T, V, \langle\rangle, 1.0, 0, \{\}$)
  Sort S by expected page savings
  Return the best m shortcuts in S

Inputs:
p *Current page in recursive traversal*
T *Trail prefix (observed page requests)*
V *Visitor identity*
T_s *Trail suffix (hypothesized pages in traversal)*
P_s *Probability of suffix* T_s
l *Length of suffix* T_s
S *Set of shortcut destinations and their savings*

ExpectedSavings(p, T, V, T_s, P_s, l, S)
  If ($l \geq$ depth bound) or ($P_s \leq$ probability threshold)
    Return S
  If ($l \leq 1$)
    CurrentSavings $\leftarrow 0$
  Else
    CurrentSavings $\leftarrow P_s \times (l-1)$
  If $p \notin S$
    Add p to S with Savings(p) = *CurrentSavings*
  Else
    Savings(p) $\leftarrow$ Savings(p) + *CurrentSavings*
  Trail $\leftarrow$ concatenate T and T_s
  For each link $p \rightarrow q$
    $P_q \leftarrow$ probability of following $p \rightarrow q$ given *Trail* and V
    $T_q \leftarrow$ concatenate T_s and $\{q\}$
    $S \leftarrow$ ExpectedSavings($q, T, V, T_q, P_q, l+1, S$)
  Return S

Table 1: **MinPath algorithm.**

end their browsing trails. Thus, the probability $P(p_i = p_{sink} | \langle p_0, \ldots, p_{i-1} \rangle, V)$ is the probability that the visitor will request no further pages in this trail. Finally, note that the models are learned offline, prior to their use by MINPATH. Only the evaluation of the model must run in real time.

3.1 Unconditional model

The simplest model of web usage predicts the next page request p_i without conditioning on any information. We learn this model by measuring the proportion of requests for each page q on the site during the training period[1]:

$$P(p_i = q) = \frac{\text{number of times } q \text{ requested}}{\text{total number of page requests}}$$

We assume the visitor can view a page only if it is linked from the current page. Thus MINPATH forces the probabilities of pages not linked from the current page to be zero and renormalizes the probabilities of the available links. If the current page is p_{i-1}, then MINPATH calculates:

$$P(p_i = q | p_{i-1}) = \begin{cases} \frac{P(p_i=q)}{\sum_{q'} P(p_i=q')} & \text{if } p_{i-1} \rightarrow q \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

where the q' are all the pages to which p_{i-1} links.

Because we have a limited volume of training data (approximately 129,000 page requests for approximately 8,000 unique URLs in a site with 240,000 web pages), we cannot build a model that predicts each and every page — many pages are requested too infrequently to reliably estimate their probability. Instead, we group pages together to increase their aggregate usage counts, and replace page requests by their corresponding group label (much in the spirit of [Zukerman *et al.*, 2000]). Specifically, we use the hierarchy that the URL directory structure imposes as a hierarchical clustering of the pages, and select only the most specific nodes (the ones closest to the leaves) that account for some minimum amount of traffic, or usage, on the site. The pages below each node share a common URL prefix, or *stem*, which we use as the label of the node. By varying the minimum usage threshold, we select more or fewer nodes; in section 4, we report how MINPATH's performance is correspondingly affected.

3.2 Naïve Bayes mixture model

The unconditional model assumes all trails on the site are similar — that a single model is sufficient to accurately capture their behavior. Common intuition suggests this assumption is false — different visitors produce different trails, and even the same visitor may follow different trails during separate visits. As an alternative, we hypothesize that each trail belongs to one (or a distribution) of K different *clusters*, each described by a separate model. We can thus compute the probability of requesting page q by conditioning on the cluster identity C_k:

$$P(p_i = q | \langle p_0, \ldots, p_{i-1} \rangle) = \sum_{k=1}^{K} P(p_i = q | C_k) P(C_k | \langle p_0, \ldots, p_{i-1} \rangle) \quad (1)$$

The result is a *mixture model* that combines the probability estimates $P(p_i = q | C_k)$ of the K different models according to a distribution over the models. By Bayes' theorem, $P(C_k | \langle p_0, \ldots, p_{i-1} \rangle) \propto P(C_k) P(\langle p_0, \ldots, p_{i-1} \rangle | C_k)$. To calculate $P(\langle p_0, \ldots, p_{i-1} \rangle | C_k)$, we make the Naïve Bayes assumption that page requests in the trail are independent given the cluster, thus: $P(\langle p_0, \ldots, p_{i-1} \rangle | C_k) = \prod_{j=0 \ldots i-1} P(p_j | C_k)$. The resulting model is a *Naïve Bayes mixture model* (similar to those used in AUTOCLASS [Cheeseman *et al.*, 1988]) for which we learn the model parameters $P(p_i = q | C_k)$ and the cluster assignment probabilities $P(C_k)$ using the EM algorithm [Dempster *et al.*, 1977].

The mixture model uses the probabilities $P(C_k | \langle p_0, \ldots, p_{i-1} \rangle)$ as a "soft" assignment of the trail to the cluster — each cluster C_k contributes fractionally in the sum in Equation 1. Alternatively, we may use a "hard" assignment of the trail to the most probable cluster, C_*. We explore both of these possibilities in section 4. The value of K may be fixed in advance, or found using holdout data. For each value of K, we compute the likelihood of the holdout

[1] More precisely, throughout our implementation we use MAP estimates with Dirichlet priors.

data given the previously learned model, and choose the K that maximizes the holdout likelihood.

An additional piece of information useful when selecting the cluster assignment is the visitor's identity, which we can incorporate by conditioning Equation 1 on V. If we assume that page requests are independent of the visitor given the cluster, then the only change to the right side of Equation 1 is that $P(C_k)$ becomes $P(C_k|V)$. Unlike an individual trail, a visitor's behavior may not be well represented by any single model in the mixture. Thus, we represent a visitor as a mixture of models, and estimate the $P(C_k|V)$ as the proportion of the visitor's history that is predicted by C_k. Specifically, let $H = \{T_1, \ldots, T_h\}$ be the set of h trails the visitor has produced on the site previous to the current visit, and $P(T_i|C_j)$ the probability that cluster C_j produced trail T_i; then $P(C_k|V) = \sum_{i=1}^{h} P(T_i|C_k)/h$.

3.3 Markov models

Both the unconditional and Naïve Bayes mixture models ignore a key piece of information from the web accesses: the sequential nature of the page trails. A *first-order Markov* model, on the other hand, incorporates this information by conditioning the probability of the next page on the previous page: $P(p_i = q|p_{i-1})$. The Markov model is trained by counting the transitions from pages p_{i-1} to p_i in the training data, and by counting how often each page appears as the initial request in a trail. As we did earlier, we replaced the URLs of page requests with URL stems to increase the volume of relevant training data. The need for this transformation is even greater for the Markov model because it has quadratically more probability values to estimate than the unconditional model, and the events (the links $p_{i-1} \rightarrow p_i$) are more rare.

In addition to a single Markov model, we also evaluate MINPATH using a mixture of Markov models [Cadez *et al.*, 2000]. We use the same EM-based method to build these mixtures as we did to learn the Naïve Bayes mixture model.

3.4 Positional and Markov/Positional models

In addition to conditioning the probability on the last requested page, we also consider conditioning on the ordinal position of the request in the visitor's trail: $P(p_i = q|i)$ or $P(p_i = q|i, p_{i-1})$. Effectively, this model is equivalent to training a separate model (either unconditional or Markov) for each position in the trail (although, for practical purposes, we treat all positions after some limit L as the same position). Visual inspection of the training trails led us to hypothesize that these models may better predict behavior, although conditioning on the additional information increases the amount of training data necessary to properly fit the model.

4 Results

We evaluate MINPATH's performance on usage at our home institution's web site. We used web access data during September 2000 to produce a training set of 35,212 trails (approximately 20 days of web usage) and a test set of 2,500 trails (approximately 1.5 days of usage); the time period from which the test trails were drawn occurred strictly after the training period. During the training and testing periods, 11,981 unique pages were requested from the total population of 243,119 unique URLs at the site. We selected only those trails with link length at least two, because shorter trails cannot be improved. We set MINPATH's link depth bound to 8 and probability threshold to 10^{-5}; in all our experiments the probability threshold proved to be the tighter bound.

We measure MINPATH's performance by the number of links a visitor must follow to reach the end of the trail. We estimate visitor behavior when provided shortcuts by making two assumptions. First, we assume that, when presented with one or more shortcuts that lead to destinations along the visitor's trail, the visitor will select the shortcut that leads farthest along the trail (*i.e.*, the visitor greedily selects the apparently best shortcut). Second, when no shortcuts lead to pages in the visitor's trail, the visitor will follow the next link in the trail (*i.e.*, the visitor will not erroneously follow a shortcut). Note, finally, that MINPATH places shortcuts on each page the visitor requests, and so the visitor may follow multiple shortcut links along a single trail.

Without shortcuts, the average length of trails in the test set is 3.42 links. Given an oracle that could predict the exact destination of the visitor's current trail, MINPATH could reduce the trail to exactly one link. The difference between 3.42 links and one link is the range of savings MINPATH can offer web visitors.

We first explored the relationship between the minimum URL usage threshold and the performance of MINPATH. We compared thresholds of 1% (which produces 42 URL stems), 0.5% (78 stems), 0.025% (1,083 stems), and 0.0% (all the unique URLs in the training data). We found that MINPATH's performance improves as we increase the number of URL stems, until the threshold falls below 0.025%. After that point, the average number of links per trail increases; we hypothesize that, because of data sparseness, we cannot learn the model as well. Thus, for all the experiments in this section, we use the best threshold we found, 0.025%. In ongoing work, we are evaluating the use of a lower threshold when MINPATH is given substantially more training data.

We next compared MINPATH's performance when using a variety of models (see Figure 1). The first column shows the number of links followed in the unmodified site. In the second and third sets of columns, MINPATH uses, respectively, an unconditional and Markov model and produces 1, 3, or 5 shortcuts. In the last two sets, MINPATH uses mixture models of either 10 or 25 clusters, and selects the distribution of the models in the mixtures based on only the current trail prefix (ignoring past visitor behavior). This graph demonstrates first that MINPATH does reduce the number of links visitors must follow — when using a mixture of Markov models and suggesting just three shortcuts, MINPATH can save an average of 0.97 links, or 40% of the possible savings. Second, we see that the Markov model, by conditioning on the sequence information, outperforms the unconditional model substantially — three shortcuts suggested with the Markov model are better than five shortcuts found with the unconditional model. Third, these results indicate the mixture models provide a slight advantage over the corresponding single model (for example, 2.72 for the Naïve Bayes mixture model versus 2.75 for the unconditional model). We computed the average of the difference in trail length between the single model and the mixture model for each test trail, and found the gains are significant at the 5% level. Finally, we found that the dif-

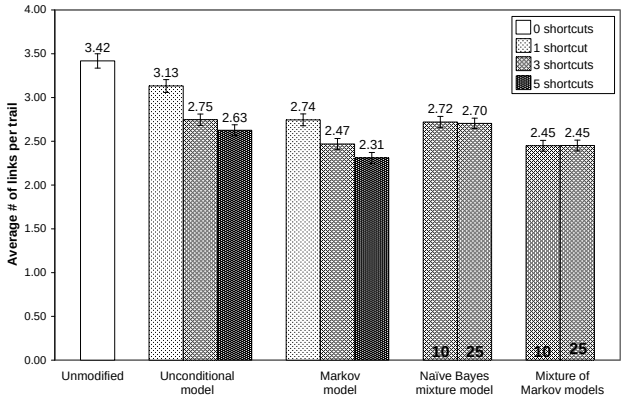


Figure 1: **MinPath's performance.** Each column shows the average number of links followed in a trail. The mixture model columns are annotated with the number of clusters. All error-bars denote 95% confidence intervals.

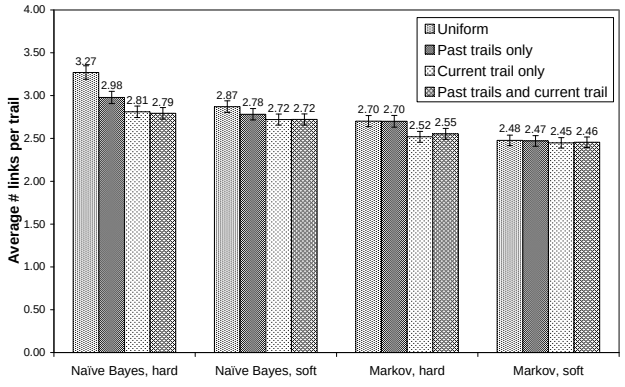


Figure 2: **Varying model assignment strategy.** Each of the four series represents a different model assignment strategy.

ferences between 10 and 25 clusters in the mixture are not statistically significant.

In Figure 2, we compare methods for selecting the mixture distribution for a trail prefix, using mixtures of 10 models. Each group of columns shows a different model and assignment type (hard or soft) combination. In each group, we condition the assignment on no information (*i.e.*, we use a uniform distribution for the soft assignment and random selection for the hard assignment), the visitor's past trails, the visitor's current trail, and both the past and current trails. Our first conclusion is that soft assignment is a better choice for both mixture models (significant at the 5% level). Second, both past trails and the current trail prefix help MINPATH select an appropriate assignment to the cluster models. However, the combination of both features is not significantly better than using just the current trail prefix with the Naïve Bayes mixture model, and does slightly worse than just the current trail with the mixture of Markov models. This result is somewhat surprising; we had expected, especially when the prefix is short, that the past trails would provide valuable information. Apparently, however, even the first one or two page requests in a trail are sufficient to assign it to the appropriate clusters. In future work we will investigate if this result remains true for larger sites.

Our last variation of models conditions the probability on the ordinal position of the page request in the trail. We compared the unconditional and Markov models against positional and Markov/positional models, choosing several values of the limit L of number of positions. In all cases, MINPATH did not perform significantly differently when using the positional information than when ignoring the position.

We finally note that MINPATH's running time is quite small. The models MINPATH uses are learned offline, but the process usually requires only several minutes. Given a model and the trail prefix, MINPATH finds a set of shortcuts in 0.65 seconds on an average desktop PC, fast enough to suggest shortcuts in real time for wireless visitors.

5 Related work

Perkowitz [2001] addresses the shortcut link problem, but uses a simpler shortcut prediction method: for each page P viewed on the site, record how often every other page Q is viewed after P in some trail. When page P is requested in the future, the shortcuts are the top m most-requested pages Q. Effectively, this approach estimates the probability that a visitor at P will eventually view Q by counting how often this event has occured in the training data. MINPATH also estimates this probability, but does so by composing the page transition probabilities along a trail through the site. The advantage of our approach is that it reduces data sparseness, although at the expense of making a first-order assumption that may not hold in practice. However, experience in other applications (*e.g.*, speech, NLP, computational biology) suggests the advantage outweighs the disadvantage. MINPATH offers two additional improvements relative to Perkowitz's approach. First, MINPATH can build more accurate models of visitor behavior by clustering visitors and building mixture models; in contrast, Perkowitz's approach builds a single shortcut table for all the visitors at the site. Second, MINPATH admits a more versatile selection of shortcuts. For example, we are currently extending MINPATH to calculate the expected savings of each shortcut given the existence of the other shortcuts added to the requested page. Perkowitz's approach cannot take advantage of this conditional information, because it derives its recommendations directly from the original usage data.

Our MINPATH algorithm shares many traits with a number of web page recommendation systems developed in recent years. Letizia [Lieberman, 1995] is a client-side agent that browses the web in tandem with the visitor. Based on the visitor's actions (*e.g.*, which links the visitor followed, whether the visitor records a page in a bookmarks file, etc.), Letizia estimates the visitor's interest in as-yet-unseen pages. Unlike MINPATH, which resides on a web server, Letizia is constrained to the resources on the web visitor's browsing device, and is thus not well suited to a wireless environment. In addition, Letizia cannot leverage the past experiences of *other* visitors to the same site — Letizia knows about the actions of only its visitor.

WebWatcher [Joachims *et al.*, 1997], Ariadne [Jühne *et al.*, 1998], and adaptive web site agents [Pazzani and Bill-

sus, 1999] are examples of web tour guides, agents that help visitors browse a site by suggesting which link each visitor should view next. With the assistance of a tour guide, visitors can follow trails frequently viewed by others and avoid becoming lost. However, tour guides assume that every page along the trail is important, and typically are limited to only suggesting which link on a page to follow next (as opposed to creating shortcuts between pages).

SurfLen [Fu *et al.*, 2000] and PageGather [Perkowitz and Etzioni, 2000] suggest pages to visit based on page requests co-occurrent in past sessions[2]. These algorithms suggest the top m pages that are most likely to co-occur with the visitor's current session, either by presenting a list of links (SurfLen) or by constructing a new index page containing the links (PageGather). However, both of these systems assume the visitor can easily navigate a lengthy list of shortcuts, and thus provide perhaps dozens of suggested links. MINPATH improves on these algorithms by factoring in the relative benefit of each shortcut, and suggesting only the few best links specific to each page request.

The predictive web usage models we present are related to previous works on sequence prediction and web usage mining. These works are too numerous to review here, but we mention two closely related ones. Most similar to our own work, WebCANVAS [Cadez *et al.*, 2000] is a system for visualizing clusters of web visitors using a mixture of Markov models. We apply similar models to web behavior, although our goal is to build predictive structures, while WebCANVAS emphasizes visualizing the clusters themselves. Sarukkai [2000] uses a Markov model of web usage to suggest the most probable links a visitor may follow, and notes the need to reduce the size of the model by clustering the URLs. Our work explores this model as well as many others, and uses the expected savings of a link, not just the link probability, to sort the resulting suggestions.

6 Conclusions

Wireless web devices will soon outnumber desktop browsers, and sites must be prepared to deliver content suited to their unique needs. Because of the high cost of navigation on a mobile device, *shortcut links* are a fruitful adaptation to augment existing content. In this paper we have made the following contributions:

- We developed the MINPATH algorithm, which finds shortcut links targeted for each web visitor's information gathering behavior;
- We explored several predictive models of web usage and evaluated how they perform with MINPATH;
- We provided empirical evidence that MINPATH can find useful shortcut links. Using a mixture of Markov models, MINPATH can save wireless visitors more than 40% of the possible link savings.

MINPATH offers many fruitful lines of continued research, and we are currently exploring several directions. One direction is studying how MINPATH will scale to larger sites, with more pages, more links between pages, and more traffic. Another is automatically selecting concise and descriptive anchor texts for shortcut links. In a third direction we are integrating MINPATH with our general framework for adapting web sites, which includes a more elaborate visitor model and incorporates the cost of adding a shortcut link and the probability of erroneously following a shortcut. Finally, we are currently conducting a user study to evaluate MINPATH on a fielded web site.

[2] A session is like a trail but relaxes the requirement of coherence in space.

References

[Anderson *et al.*, 2001] C. R. Anderson, P. Domingos, and D. S. Weld. Personalizing web sites for mobile users. In *Proc. 10th Intl. WWW Conf.*, 2001.

[Cadez *et al.*, 2000] I. V. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of navigation patterns on a web site using model based clustering. In *Proc. 6th Intl. Conf. on Knowledge Discovery and Data Mining*, 2000.

[Cheeseman *et al.*, 1988] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. AutoClass: A Bayesian classification system. In *Proc. 5th Intl. Conf. on Machine Learning*, 1988.

[Dempster *et al.*, 1977] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc., Ser. B*, 39(1):1–38, 1977.

[Fink *et al.*, 1996] J. Fink, A. Kobsa, and A. Nill. User-oriented Adaptivity and Adaptability in the AVANTI Project. In *Designing for the Web: Empirical Studies*, Microsoft Usability Group, 1996.

[Fu *et al.*, 2000] X. Fu, J. Budzik, and K. J. Hammond. Mining navigation history for recommendation. In *Proc. 2000 Conf. on Intelligent User Interfaces*, 2000.

[Joachims *et al.*, 1997] T. Joachims, D. Freitag, and T. Mitchell. WebWatcher: A tour guide for the World Wide Web. In *Proc. 15th Intl. Joint Conf. on Art. Int.*, 1997.

[Jühne *et al.*, 1998] J. Jühne, A. T. Jensen, and K. Grønbæk. Ariadne: a Java-based guided tour system for the World Wide Web. In *Proc. 7th Intl. WWW Conf.*, 1998.

[Lieberman, 1995] H. Lieberman. Letizia: An agent that assists web browsing. In *Proc. 14th Intl. Joint Conf. on Art. Int.*, 1995.

[Pazzani and Billsus, 1999] M. J. Pazzani and D. Billsus. Adaptive web site agents. In *Proc. 3rd Intl. Conf. on Autonomous Agents*, 1999.

[Perkowitz and Etzioni, 1997] M. Perkowitz and O. Etzioni. Adaptive web sites: an AI challenge. In *Proc. 15th Intl. Joint Conf. on Art. Int.*, 1997.

[Perkowitz and Etzioni, 2000] M. Perkowitz and O. Etzioni. Towards adaptive web sites: Conceptual framework and case study. *Art. Int. J.*, 118(1–2), 2000.

[Perkowitz, 2001] M. Perkowitz. *Adaptive Web Sites: Cluster Mining and Conceptual Clustering for Index Page Synthesis*. PhD thesis, Dept. of Comp. Sci. and Eng., Univ. of Washington, 2001.

[Sarukkai, 2000] R. R. Sarukkai. Link prediction and path analysis using Markov chains. In *Proc. 9th Intl. WWW Conf.*, 2000.

[Wexelblat and Maes, 1999] A. Wexelblat and P. Maes. Footprints: History-rich tools for information foraging. In *Proc. ACM CHI 1999 Conf. on Human Factors in Comp. Sys.*, 1999.

[Yan *et al.*, 1996] T. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal. From user access patters to dynamic hypertext linking. In *Proc. 5th Intl. WWW Conf.*, 1996.

[Zukerman *et al.*, 2000] I. Zukerman, D. Albrecht, A. Nicholson, and K. Doktor. Trading off granularity against complexity in predictive models for complex domains. In *Proc. 6th Intl. Pacific Rim Conf. on Art. Int.*, 2000.

Using Text Classifiers for Numerical Classification

Sofus A. Macskassy, Haym Hirsh, Arunava Banerjee, Aynur A. Dayanik
{sofmac,arunava,aynur,hirsh}@cs.rutgers.edu
Department of Computer Science
Rutgers University
110 Frelinghuysen Rd
Piscataway, NJ 08854-8019

Abstract

Consider a supervised learning problem in which examples contain both numerical- and text-valued features. To use traditional feature-vector-based learning methods, one could treat the presence or absence of a word as a Boolean feature and use these binary-valued features together with the numerical features. However, the use of a text-classification system on this is a bit more problematic —in the most straight-forward approach each number would be considered a distinct token and treated as a word. This paper presents an alternative approach for the use of text classification methods for supervised learning problems with numerical-valued features in which the numerical features are converted into bag-of-words features, thereby making them directly usable by text classification methods. We show that even on purely numerical-valued data the results of text-classification on the derived text-like representation outperforms the more naive numbers-as-tokens representation and, more importantly, is competitive with mature numerical classification methods such as C4.5 and Ripper.

1 Introduction

The machine learning community has spent many years developing robust classifier-learning methods, with C4.5[Quinlan, 1993] and Ripper [Cohen, 1995] two popular examples of such methods. Although for many years the focus has been on numerical and discrete-valued classification tasks, over the last decade there has also been considerable attention to text-classification problems[Sebastiani, 1999; Yang, 1999]. Typically such methods are applied by treating the presence or absence of each word as a separate Boolean feature. This is commonly performed either directly, by generating a large number of such features, one for each word, or indirectly, by the use of set-valued features [Cohen, 1996], in which each text-valued field of the examples is viewed as a single feature whose value for an example is the set of words that are present in that field for this example.

The information retrieval community has similarly spent many years developing robust retrieval methods applicable to many retrieval tasks concerning text-containing documents, with vector-space methods [Salton, 1991] being the best known examples of techniques in this area. Although for many years the focus has been primarily on retrieval tasks, here, too, the last decade has seen a significant increase in interest in the use of such methods for text-classification tasks. The most common techniques use the retrieval engine as the basis for a distance metric between examples, for either direct use with nearest-neighbor methods [Yang and Chute, 1994], or, based on the closely related Rocchio [1971] relevance feedback technique, for use after creating a summary "document" for each class and retrieving the nearest one [Schapire *et al.*, 1998].

The irony is that although we now have much experience on placing text-clasification problems in the realm of numerical-classification methods, little attention has been brought to the question of whether numerical-classification problems can be effectively brought into the realm of text-classification methods. Since the text-retrieval methods on which they are based have many decade's maturity, if done effectively they have the potential of broadening further our base of methods for numerical classification.

More importantly for us, however, is the fact that many real-world problems involve a combination of both text- and numerical-valued features. For example, we came to ask these questions by confronting the problem of email classification, where we wanted to explore instance-representations that considered not only the text of each message, but also the length of the message or the time of day at which it is received [Macskassy *et al.*, 1999]. Although the machine-learning-derived methods that we now know how to apply to pure text-classification problems could be directly applied to these "mixed-mode" problems, the application of information-retrieval-based classification methods was more problematic. The most straight-forward approach is to treat each number that a feature may take on as a distinct "word", and proceed with the use of a text-classification method using the combination of true words and tokens-for-numbers words. The problem is that this makes the numbers 1 and 2 as dissimilar as the numbers 1 and 1000 —all three values are unrelated tokens to the classification method. What we would like is an approach to applying text-classification methods to problems with numerical-valued features so that the distance between such numerical values is able to be discerned by the classification method.

This paper presents one way to do just this, converting numerical features into features to which information-retrieval-based text-classification methods can apply. Our approach presumes the use of text-classification methods that treat a piece of text as a "bag of words", representing a text object by an unordered set of tokens present in the text (most commonly words, but occasionally tokens of a more complex derivation).

The core of our approach is, roughly, to convert each number into a bag of tokens such that two numbers that are close together have more tokens in common in their respective bags than would two numbers that are farther apart. The high-level idea is to create a set of landmark values for each numerical feature, and assign two tokens to each such landmark. Every value of a feature for an example will be compared to each

landmark for that feature. If the example's value is less than or equal to the landmark, the first of that landmark's two tokens is placed in that example's "bag". If the value is more than the landmark the second token is instead used in the bag. The result is that every feature gets converted into a bag of tokens, with each bag containing the same number of entries, each differing only to reflect on which side of each landmark a value lies.[1]

The key question then becomes how to pick landmark values. Although our experiments show that even fairly naive approaches for selecting landmarks can perform quite well, we instead appeal to the body of work on feature discretization that has already been well-studied within machine learning [Catlett, 1991; Kerber, 1992; Fayyad and Irani, 1993; Dougherty *et al.*, 1995; Kohavi and Sahami, 1996; Frank and Witten, 1999]. Learning methods such as C4.5 would normally have to consider a large number of possible tests on each numerical feature, in the worst case one between each consecutive pair of values that a feature takes on. These discretization methods instead use a variety of heuristic means to identify a more modest —and hence more tractable —subset of tests to consider during the learning process. Our approach is thus to apply such methods to identify a set of landmark values for each numerical feature and create two possible tokens for each landmark value, exactly one of which is assigned to each example for each landmark. The result is a "bag of words" representation for each example that can then be used by text-classification methods.

In the remainder of this paper we first describe our approach for converting numerical features into bag-of-words features in more detail, including the landmark-selection methods that we used. We then describe our experimental evaluation of our approach: the learning methods, evaluation methods used, and our results — which show that the text-classification methods using our bag-of-words representation perform competitively with the well-used methods C4.5 and Ripper when applied to the original numerical data. We conclude the paper with some analysis of these results and some final remarks.

2 Converting Numbers to Bags of Tokens

The approach taken in this paper is to convert every number into a set of tokens such that if two values are close, these sets will be similar, and if the values are further apart the sets will be less similar. This is done for each feature by finding a set of "landmark values" or "split-points" within the feature's range of legitimate values by analyzing the values that the feature is observed to take on among the training examples. Given an example, its numerical value for a given feature is compared to each split-point for that feature generated by our approach, and for each such comparison a token will be added, representing either that the value is less than or equal to the particular split-point or greater than that split-point. This will result in exactly one token being added per split-point.

For example, consider a news-story classification task that includes a numerical feature representing the story's length. We can artificially invent a set of split-points to demonstrate our process, such as 500, 1500, and 4000. For each split-point we define two tokens, one for either side of the split-point a value may lie. Using the above split-points for the length of a news-story would result in the tokens "lengthunder500", "lengthover500", "lengthunder1500", "lengthover1500", "lengthunder4000", and "lengthover4000". A new message of length 3000 would thereby have its length feature converted into the set of tokens "lengthover500", "lengthover1500", and "lengthunder4000". These would be added to the bag-of-words representation of the example — whether the other words in the bag were created from other numerical features, or the result of pre-existing text-valued features. More abstractly, consider the hypothetical numerical feature plotted along a number line in Figure 1. If a training example is obtained whose value for this feature falls in bin2, the set {morethansplit1, lessthansplit2, lessthansplit3, lessthansplit4} would be the bag-of-words representation created for this value. Note that more than one value can be given the same representation, as long as they all fall between the same two consecutive split-points.

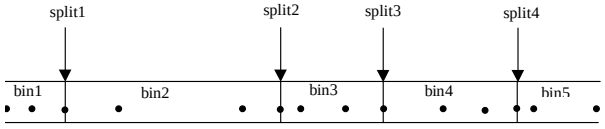


Figure 1: An example feature range and construction of bins. Dots represent values and rectangles represent bins.

The key question, of course, is how these split-points are selected. We use two methods in our main results. The first, called the *MDL* method, uses an existing entropy-based method for discretization to find good split-points [Fayyad and Irani, 1993]. This method is very similar to one that uses C4.5 on the training data, restricting it to ignore all but the single feature for which split-points are being selected, harvesting the decision points found within each of the internal nodes of the tree [Kohavi and Sahami, 1996]. The second method, which we call the *density* method, selects split-points that yield equal-density bins —where the number of values that fall between consecutive split-points stays roughly constant. The number of bins is selected using hill-climbing on error rate using a hold-out set. In the rest of this section we describe these two methods in more detail, concluding with a discussion of how we handle examples that are missing values for one or more of their features.

2.1 The MDL Method

The MDL method [Fayyad and Irani, 1993; Kohavi and Sahami, 1996] makes use of information-theoretic techniques to analyse the values of a numeric feature and create split-points that have high information gain. It does so in a recursive fashion, finding one split-point in the overall set of values, then finding another split-point in each of the two created subsets, until a stopping criteria based on Minimum Description Length priciples [Rissanen, 1987] is met. Each numerical feature is treated separately, yielding a different set of split-points for each feature. Due to space limitations we refer the reader to the given citations for further details about this popular discretization algorithm.

2.2 The Density Method

The density method (described in algorithmic form in Figure 2) begins by obtaining and sorting all values observed in the data for a feature f, yielding an ordered list S_f. Thus, for example, the result for some feature f might be $S_f = \langle 1, 2, 2, 2, 5, 8, 8, 8, 9 \rangle$. Given some desired number of splits k, the density method splits the feature set S_f into $k+1$ sets of equal size (except when rounding effects require being off by one) such that split-point s_j is the $(|S_f| \times \lfloor \frac{j}{k+1} \rfloor)$-th point in S_f. Using this split-point, two tokens are generated; one for when a numerical value is less than or equal to the split-point, and another for when the numerical value is greater than the

[1] However, as we will explain later, there may be fewer values in case of missing values.

Inputs: Sets of values for each numeric feature.
Algorithm:
$currError \leftarrow 100$ /* assume errors run from $0 - 100$ */
$lastError \leftarrow 100$
$numSplits \leftarrow 1$
$maxSplits \leftarrow \text{argmax}_{f \in \text{numerical features}}(|S_f|)$
while($numSplits < maxSplits$) do
  for each numerical feature f
    $n_f \leftarrow \min(numSplits, |S_f|)$
    /* Create n_f split-points for feature f. */
    Use as split-points for f the j-th element of S_f
      for $j = |S_f| \times \lfloor \frac{i}{n_f+1} \rfloor$ with i running from 1 to n_f
  end
  Divide data into 70% for $train$ and 30% for $test$.
  $C \leftarrow$ Run learner on $train$ with current split-points.
  $currError \leftarrow$ Evaluate C on $test$.
  if($currError = 0$) do
    $maxSplits \leftarrow 0$
    $lastError \leftarrow currError$
  else if($lastError < currError$) do
    $numSplits \leftarrow numSplits/2$
    $maxSplits \leftarrow 0$
  else
    $numSplits \leftarrow 2 \times numSplits$
    $lastError \leftarrow currError$
  end
end
Outputs: $lastError$

Figure 2: Density algorithm for finding split-points.

split-point. k —the final number of split-points —is found using a global hill-climbing search with the number of split-points growing geometrically by a factor of two until the observed error rate on a 30% hold-out set is observed to increase or reach 0.

One final detail of the density method is that the algorithm in Figure 2 is actually run twice. The first time is based on a "volumetric" density calculation, where duplicate values are included in the analysis. After doing this the algorithm is run a second time after duplicate values have been removed, yielding a second set of split points that have an (approximately) equal number of *distinct* values between each split-point. Thus, for example, for the feature f this second run would now use the list $S_f = \langle 1, 2, 5, 8, 9 \rangle$. Whichever method yielded a lower error rate gives the final set of split points. If they have the same performance, whichever had fewer split-points is selected.

2.3 Missing Values

A common occurrence for many learning problems is when some examples are missing values for some of the features. This can be a complicating factor both during the learning phase, when assessing the importance of features in forming some learning result, and in classification, when making a decision when values of some of the attributes are unavailable. Common approaches range from simply deleting data or features to remove such occurrences, to imputing some value for the feature —such as through learning or through something as simple as using the median, mean, or mode value in the training data, to more complex methods such as are used in learning algorithms such as C4.5. Our approach for creating bag-of-word features out of numerical features contributes another interesting way to handle missing values. The idea is, quite simply, to add no tokens for a feature of an example when the value for this feature is missing. By not committing to any of the tokens that this feature might otherwise have added it neither contributes nor detracts from the classification process, allowing it to rely on the remaining features in assigning a label to the example.

3 Learning Algorithms

In this section we briefly describe the learning algorithms that we use in our experiments. Recall that our goal is to demonstrate that, using our approach, text-classification methods can perform as credibly on numerical classification problems as more traditional methods that were explicitly crafted for such problems. To show this we use a sampling of four different approaches for text classification. Our first is based on the Rocchio-based vector-space method for text retrieval mentioned earlier, which we label TFIDF [Joachims, 1997; Schapire *et al.*, 1998; Sebastiani, 1999]. We also consider two probabilistic classification methods that have become popular for text classification, Naive Bayes [Domingos and Pazzani, 1996; Joachims, 1997; Mitchell, 1997] and Maximum Entropy [Nigam *et al.*, 1999]. Finally, we also use the Ripper rule learning system [Cohen, 1995; 1996], using its capability of handling set-valued features so as to handle text classification problems in a fairly direct fashion. (The first three methods used the implementations available as part of the *Rainbow* text-classification system[McCallum, 1996].) The baselines to which we compare the text-classification methods are two popular "off the shelf" learning methods, C4.5 (release 8) [Quinlan, 1993] and Ripper. Note that Ripper has been mentioned twice, once as a text-classification method using our transformed features encoded for Ripper as set-valued features, and the other using the original numerical features in the same fashion as they are used with C4.5. Thus Ripper is in the unique position of being both a text-classification method when used with one representation, and as a numerical classification method when used with the other. Missing values were handled for the text-classification methods as discussed earlier, by simply not generating any tokens for a feature of an example when it had no given value, and for C4.5 and Ripper (when used with numerical features) by their built-in techniques for handling missing values.

The *TFIDF* classifier is based on the relevance feedback algorithm by Rocchio [1971] using the vector space retrieval model. This algorithm represents documents as vectors so that documents with similar content have similar vectors. Each component of such a vector corresponds to a term in the document, typically a word. The weight of each component is computed using the TFIDF weighting scheme, which tries to reward words that occur many times but in few documents. In the learning phase, a prototype vector is formed for each class from the positive and negative examples of that class. To classify a new document d, the cosines of the prototype vectors with the corresponding document vector are calculated for each class. d is assigned to the class with which its document vector has the highest cosine.

Naive Bayes is a probabilistic approach to inductive learning. It estimates the a posteriori probability that an example belongs to a class given the observed feature values of the example, assuming independence of the features. The class with the maximum a posteriori probability is assigned to the example. The naive Bayes classifier used here is specifically designed for text classification problems.

The *Maximum Entropy* classifier (labeled MAXENT in our results) estimates the conditional distribution of the class label given a document, which is a set of word-count features. The high-level idea of this technique is, roughly, that uniform dis-

tributions should be prefered in the absence of external knowledge. A set of constraints for the model are derived from the labeled training data, which are expected values of the features. These constraints characterize the class-specific expectations for the model distribution and may lead to minimal non-uniform distributions. The solution to the maximum entropy formulation is found by the improved iterative scaling algorithm [Nigam *et al.*, 1999].

Ripper is a learning method that forms sets of rules, where each rule tests a conjunction of conditions on feature values. Rules are returned as an ordered list, and the first successful rule provides the prediction for the class label of a new example. Importantly, Ripper allows attributes that take on sets as values, in addition to numeric and nominal features, and a condition can test whether a particular item is part of the value that the attribute takes on for a given example. This was designed to make Ripper particularly convenient to use on text data, where rather than listing each word as a separate feature, a single set-valued feature that contains all of an instance's words is used instead. Rules are formed in a greedy fashion, with each rule being built by adding conditions one at a time, using an information-theoretic metric that rewards tests that cause a rule to exclude additional negative data while still hopefully covering many positive examples. New rules are formed until a sufficient amount of the data has been covered. A final pruning stage adjusts the rule set in light of the resulting performance of the full set of rules on the data.

C4.5 is a widely used decision tree learning algorithm. It uses a fixed sets of attributes, and creates a decision tree to classify an instance into a fixed set of class-labels. At every step, if the remaining instances are all of the same class, it predicts that class, otherwise, it chooses the attribute with the highest *information gain* and creates a decision based on that attribute to split the training set into one subset per discrete value of the feature, or two subsets based on a threshold-comparison for continuous features. It recursively does this until all nodes are final, or a certain user-specified threshold is met. Once the decision tree is built, C4.5 prunes the tree to avoid overfitting, again based on a user-specified setting.

4 Evaluation Methodology

To compare our text-like encoding of numbers when used with text-classification systems to the use of C4.5 and Ripper on the original numerical features we used 23 data sets taken from the UCI repository [Blake and Merz, 1998]. Table 1 shows the characteristics of these datasets. The first 14 represent problems where all the features are numeric. The final 9 represent problems in which the designated number of features are numeric and the rest are discrete or binary-valued.

The accuracy of a learner was done through ten-fold stratified cross-validation[Kohavi, 1995]. Each dataset was represented in one of four ways for our experiments:

- The original feature encoding —using numbers —for use with C4.5 and Ripper.
- The bag-of-words encoding generated by the density method, for use with the four text-classifications.
- The bag-of-words encoding generated by the MDL method, for use with the four text-classifications.
- The bag-of-words encoding generated using the tokens-for-numbers approach, for use with the five text-classifications. This was accomplished by converting every number into its English words – for example, "5" becomes "five" and "2.3" becomes "twopointthree".

The first of these represents our baseline, using a machine learning method designed for numerical classification. The next two are the new approaches presented in this paper. The final one is the representation that simply treats each number as a distinct "word" without regard to its value.

Dataset	# of Instances	# of Features	# of Numeric Features	# of Classes	Base Accuracy (%)
bcancerw	699	10	10	2	66
diabetes	768	8	8	2	65
glass	214	9	9	6	36
hungarian	294	13	13	2	64
ionosphere	351	34	34	2	64
iris	150	4	4	3	33
liver	345	6	6	2	58
musk	476	166	166	2	57
new-thyroid	215	5	5	5	70
page-blocks	5473	10	10	5	90
segmentation	2310	19	19	7	14
sonar	208	60	60	2	53
vehicle	846	18	18	4	26
wine	178	13	13	2	40
arrhythmia	452	279	206	16	54
autos	205	26	16	2	88
cleveland	303	13	6	2	54
credit-app	690	15	6	2	56
cylinder-bands	512	39	20	2	61
echocardiogram[1]	132	13	9	2	44
horse	368	22	7	2	63
post-operative	90	8	1	3	71
sponge	76	45	3	3	92

[1] The MDL approach was unable to find any split-points for this data-set, so it is omitted in any comparisons on the MDL method.

Table 1: Properties of all datasets.

5 Results

The first question we ask is the key one: To what extent does our approach yield a competitive learning method on numerical classification problems? Figure 3 shows the results of comparing the four text-learning methods using our MDL-algorithm features to the two numerical classification methods. Each point represents a single data set, where the x-axis is the accuracy of either C4.5 or Ripper and the y-axis is the accuracy of one of the four text methods. Points above the y=x line represent cases where the numerical-classification method was inferior to our use of a text-classification method, and points below the line are cases where the numerical method was superior. The qualitative flavor of this graph is that the MDL-algorithm features allows text-classification methods to perform credibly in many cases, exceeding numerical methods in some cases, although performing less successfully in many cases as well. We plot in Figure 4 a similar graph comparing the four text methods using the density-algorithm features to the two numerical methods. (The "outlier" cases at the bottom of the graphs are for the post-operative data set, which has only 90 examples and only 1 of its 8 features being numeric.)

Since the preceding graphs collapse eight different comparisons (four text methods versus two numerical methods)

	TFIDF	NB	MAXENT	Ripper
MDL/C4.5	9/13/0	11/10/1	12/10/0	7/14/1
MDL/Ripper	8/14/0	11/9/2	11/11/0	8/12/2
Density/C4.5	4/19/0	7/15/1	13/10/0	9/13/1
Density/Ripper	4/19/0	8/14/1	13/9/1	8/13/2

Table 2: Comparing the number of wins/losses/ties for each featurization (MDL first two rows, density second two rows) when coupled with one of the four text-classification methods labeling the columns, versus a numerical method.

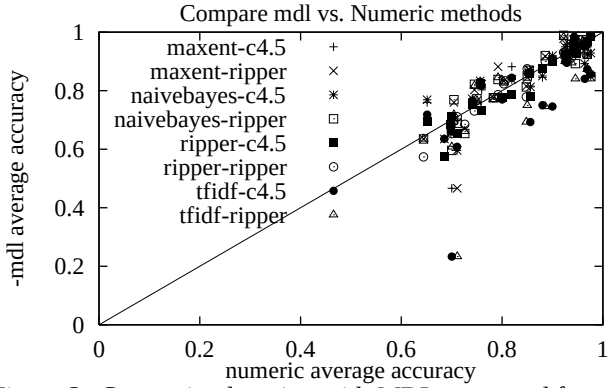


Figure 3: Comparing learning with MDL-generated features to numeric learning.

into a picture Table 2 also shows for how many data sets each text method beat a numerical method. Each entry in the table is the number of wins/losses/ties for the new featurization method used with a text method compared to a numerical classification method. The columns label the text method used. The first two rows are results when the MDL method is used, the next two are for the density method, in each case the first comparison is to numerical classification with C4.5 comes in the first of the two rows, followed by Ripper. These results show that in a non-trivial number of cases the use of our approach for converting numerical features into text-based data beats out the popular learning methods, with absolutle improvements in accuracy ranging as high as 12%. While these results do *not* show that the approach is unconditionally superior to numerical classification, they do show that the approach does merit consideration for use as a numerical classification method.

The next question we ask is whether the use of two different featurization methods is necessary: Does either dominate the other? Figure /reffig:mdlvsdens shows the results of such a comparison, where each point represents a single data set and a text learning method, with the x-axis representing the result of using the MDL method with that learning algorithm, and the y-axis representing the result of using the density method with that learning algorithm. Here, too, the results show that neither method is clearly superior, with perhaps a bit better performance in general by the MDL method, but with some cases still going to the density method.

6 Additional Analysis

We began the paper stating that the obvious approach to converting numbers into text-based feature is convert each number into a unique token, to be added as-is to the set of words

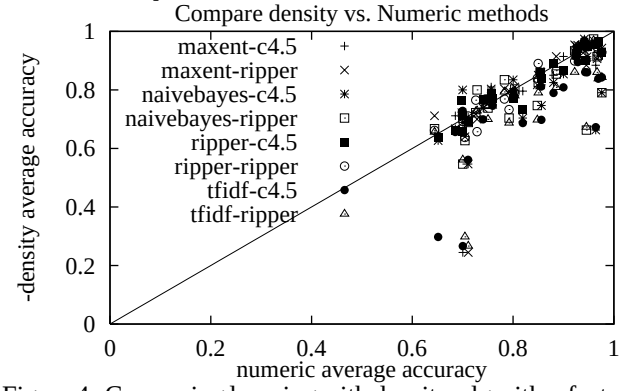


Figure 4: Comparing learning with density-algorithm features to numeric learning.

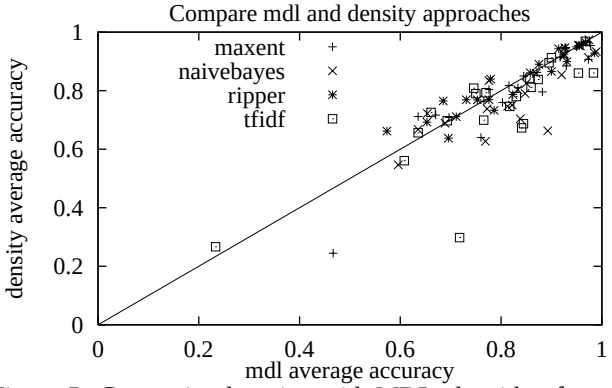


Figure 5: Comparing learning with MDL-algorithm features to density-algorithm features.

for a given example. One additional question we can ask is whether the complexity of our methods are necessary: Perhaps this simple tokenization approach performs as effectively? Figure 6 shows an analysis of this question. Each point is a data set, with the x-axis value representing the accuracy of the tokenization approach with a particular text-classification method, and the y-axis represents the accuracy with one of our two featurization methods using the same learning method. As is clearly evident from the figure, the tokenization approach is not as effective in general as our more sophisticated approach.

We conclude this section by noting that Kohavi and Sahami [1996] discuss a different discretization method that is very similar to the MDL method. This method simply runs C4.5 on the data, ignoring all features except the one for which split-points are being created. Kohavi and Sahami show that this method is slightly inferior to the MDL approach. However, just because it is inferior for discretization for decision-tree learning does not imply that it must be the case here, too. To test this we compared the four text classification methods using the MDL method to the C4.5 method. Figure 7 shows the results of this experiment. Each point represents a data set and a learning method. The x-axis represents the accuracy of the C4.5 approach, and the y-axis represents the accuracy of the MDL approach. As is clear, although there is some difference between the performance of the methods, they are somewhat similar in behavior.

7 Final Remarks

This paper has described an approach for converting numeric features into a representation enabling the application of text-classification methods to problems that have traditionally been solved solely using numerical-classification meth-

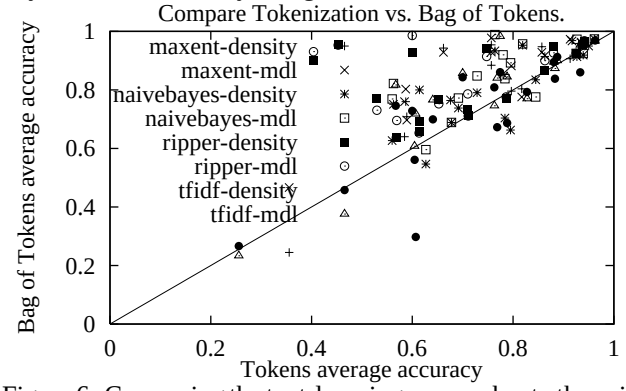


Figure 6: Comparing the text-learning approaches to the naive tokenization approach.

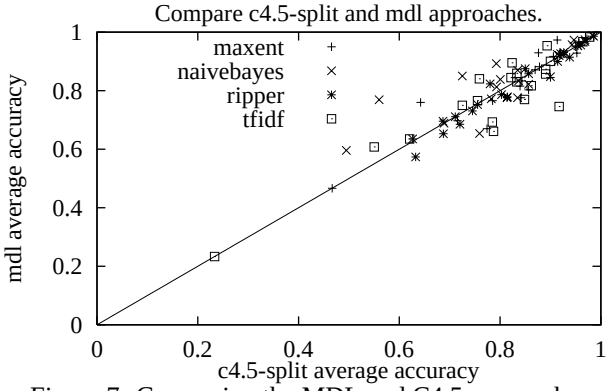


Figure 7: Comparing the MDL and C4.5 approaches.

ods. In addition to opening up the use of text-methods to problems that involve "mixed-mode" data —both numerical- and text-valued features —it yields a new approach to numerical-classification method in its own right. Our experiments show that in a non-trivial number of cases the resulting methods outperform highly optimized numerical-classification methods. Also importantly, our experiments show that our approach yields a vast improvement over the naive method of converting a numeric into its equivalent textual token.

There are many directions in which we are now taking this work. Our original motivation for performing this work was to broaden the class of learning methods that can be applied to mixed-mode data. Now that we have done so, we can return to some of the work that motivated this, performing additional evaluations of this work on mixed-mode data. Doing so, however, requires a set of benchmark problems, something that does not presently exist. We are therefore in the process of creating such data sets so we can perform this evaluation process. We also noted that our approach yields an intriguing way to deal with data with missing values, and understanding its benefits and liabilities compared to other approaches remains a question that we hope to explore. Finally, as is usually the case when comparing any two learning methods that are successful in competing cases, it is difficult to make any definitive statements about when they each may be successful. Various conjectures include differences in the amount of missing values in the different data sets, the number of numeric versus non-numeric features, etc. For the given data sets we were unable to discern any such pattern in our results. This remains an important question that we also plan to study.

Acknowledgments

We would like to thank Foster Provost, Lyle Ungar, and members of the Rutgers Machine Learning Research Group for helpful comments and discussions.

References

[Blake and Merz, 1998] C. L. Blake and C. J. Merz. Uci repository of machine learning databases, 1998.

[Catlett, 1991] J. Catlett. On changing continuous attributes into ordered discrete attributes. In Y. Kodratoff, editor, *Proceedings of the European Working Session on Learning*, pages 164–178. Berling: Springer-Verlag, 1991.

[Cohen, 1995] W. W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, Lake Tahoe, California, 1995.

[Cohen, 1996] W. W. Cohen. Learning trees and rules with set–valued features. In *AAAI96*, 1996.

[Domingos and Pazzani, 1996] P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of simple bayesian classifier. In *Proceedings of the 13th International Conference on Machine Learning*, pages 105–112, 1996.

[Dougherty *et al.*, 1995] K. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann, 1995.

[Fayyad and Irani, 1993] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on AI*, pages 1022–1027. Morgan Kaufmann, 1993.

[Frank and Witten, 1999] E. Frank and I. H. Witten. Making better use of global discretization. In *Proceedings of the 17th International Conference on Machine Learning*, Slovenia, 1999.

[Joachims, 1997] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

[Kerber, 1992] R. Kerber. Discretization of numeric attributes. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 123–128, Menlo Park, CA, 1992. AAAI Press/MIT Press.

[Kohavi and Sahami, 1996] R. Kohavi and M. Sahami. Error-based and entropy-based discretization of continuous features. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 114–119, Menlo Park, CA, 1996. AAAI Press/MIT Press.

[Kohavi, 1995] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1137–1143, San Francisco, CA, 1995. Morgan Kaufmann.

[Macskassy *et al.*, 1999] S. A. Macskassy, A. A. Dayanik, and H. Hirsh. Emailvalet: Learning user preferences for wireless email. In *Proceedings of Learning about Users Workshop, IJCAI'99*, Stockholm, Sweden, 1999.

[McCallum, 1996] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. `http://www.cs.cmu.edu/~mccallum/bow`, 1996.

[Mitchell, 1997] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[Nigam *et al.*, 1999] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *Proceedings of Machine Learning for Information Filtering Workshop, IJCAI'99*, Stockholm, Sweden, 1999.

[Quinlan, 1993] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[Rissanen, 1987] I. Rissanen. Minimum description length principle. *Encyclopedia of Statistical Sciences*, 5:523–527, 1987.

[Rocchio, 1971] J. Rocchio. Relevance feedback in information retrieval. In Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 14, pages 313–323. Prentice–Hall, 1971.

[Salton, 1991] G. Salton. Developments in automatic text retrieval. *Science*, 253:974–979, 1991.

[Schapire *et al.*, 1998] R. Schapire, Y. Singer, and A. Singal. Boosting and rocchio applied to text filtering. In *Proceedings of ACM SIGIR*, pages 215–223, 1998.

[Sebastiani, 1999] F. Sebastiani. Machine learning in automated text categorisation: a survey. Technical Report IEI-B4-31-1999, Istituto di Elaborazione dell'Informazione, 1999.

[Yang and Chute, 1994] Y. Yang and C. Chute. An example-based mapping method for text classification and retrieval. *ACM Transactions on Information Systems*, 12(3):252–277, 1994.

[Yang, 1999] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2):67–88, 1999.

Faster Association Rules for Multiple Relations

Siegfried Nijssen and **Joost Kok**
snijssen@liacs.nl and joost@liacs.nl
Leiden Institute of Advanced Computer Science, Leiden University
Niels Bohrweg 1, 2333CA Leiden, The Netherlands

Abstract

Several algorithms have already been implemented which combine association rules with first order logic formulas. Although this resulted in several usable algorithms, little attention was payed until recently to the efficiency of these algorithms. In this paper we present some new ideas to turn one important intermediate step in the process of discovering such rules, i.e. the discovery of frequent item sets, more efficient. Using an implementation that we coined FARMER, we show that indeed a speed-up is obtained and that, using these ideas, the performance is much more comparable to original association rule algorithms.

1 Introduction

The formalism of association rules was introduced by Agrawal [1996] for the purpose of basket analysis. An important step in the discovery of such rules is the construction of frequent item sets. These are, for instance, sets of items that are frequently bought together in one supermarket transaction. As this discovery step is time critical, it is obligatory that it is performed reasonably fast. Much research has been done in order to develop efficient algorithms. A well-known algorithm resulting from this research is APRIORI, of which many variants have been developed, such as APRIORI-TID [Agrawal *et al.*, 1996] and a breadth-first algorithm introduced by Pijls and Bioch [1999].

On the other hand, efforts have been done to extend the usability of association rules beyond the basic case of basket analysis. Dehaspe and De Raedt [1997] use the notion of atom sets as a first order logic extension of item sets. The incorporation of techniques from Inductive Logic Programming allows for more complex rules to be found which also take into account background knowledge. Consequently, this also allows data mining of data which is spread over tables which can not reasonably be merged into one table. An algorithm was implemented based on this notion, which was called WARMR. The usefulness of this algorithm was demonstrated in several real-world situations (see, for example, [Dehaspe *et al.*, 1998]). These experiments, however, also showed the major shortcoming of the algorithm: its efficiency proved to be very low, some experiments even taking several days.

We propose to obtain a gain in efficiency by tackling two properties of the WARMR algorithm:

- while still using the first order logic notation, we remove the need for PROLOG;
- by using a more sophisticated datastructure borrowed from an implementation of APRIORI, our algorithm does not depend on a time consuming test for equivalence.

The algorithm that we introduce has some ressemblance with the algorithm that was developped in [Blockeel *et al.*, 2000]. That algorithm however did not tackle one of the most time consuming steps of WARMR: a test for equivalence under θ-subsumption. Our algorithm pays special attention to this step and offers an alternative solution. Under some restrictions we will show that our algorithm is equivalent to WARMR. Experiments with our algorithm then show a considerable speed-up compared to WARMR.

The paper is organized as follows. In the second section we summarize the association rule algorithm on which our work is based. In the third section we discuss some important notions introduced in the WARMR algorithm. The fourth section introduces our modifications, which are verified by giving results of experiments in the fifth section. The sixth section concludes.

2 Breadth-first APRIORI

Our algorithm is based on a variation of APRIORI that was introduced by Pijls and Bioch [1999]. The algorithm performs the same task as APRIORI. Given a database D which contains subsets T of a set of items $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ the algorithm discovers all *frequent item sets*, which are the subsets $I \subseteq \mathcal{I}$ for which $support(I) = |\{T \in D \mid I \subseteq T\}|$ exceeds a predefined threshold. An item set of size k is called a $k-$item set. An important property of $support(I)$ is:

$$I_1 \subseteq I_2 \Rightarrow support(I_1) \geq support(I_2), \qquad (1)$$

as every subset I_1 of I_2 occurs in every transaction that contains I_2. This property turns an efficient bottom-up levelwise search possible: it can *apriori* be determined that a $k+1$-item set is not frequent if a k-subset is infrequent.

The breadth first-algorithm is such a bottom-up levelwise algorithm. It starts with candidates of size one, after which a process is repeated of counting $k-$candidate item sets and of using them to obtain candidate $k+1$ item sets. All these

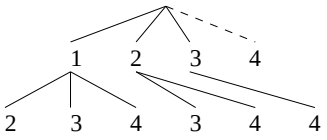


Figure 1: A small example of a trie datastructure

steps are performed on a *trie* datastructure, of which Figure 1 displays an example. Every path from the root to a node corresponds to an item set; all leafs at the deepest level correspond to candidate item sets. Paths which do not reach until the deepest level are maintained only when they correspond to frequent item sets and are displayed by dotted lines for the sake of clarity. The trie is used in the following fashion:

- In the step of candidate counting, a tree traversal is performed for every transaction, as follows: if an item occurs in a transaction, all its children are checked recursively. If a leaf is reached, the support count of the corresponding item set is increased.
- In the step of candidate generation, for every frequent item set new children are generated, consisting of all frequent right brothers. In the example $\{2\}$ is expanded by its frequent right brothers 3 and 4. This copying mechanism in combination with the order of the items takes care of generating every item set at most once.

In both steps, this mechanism distinguishes itself from the original APRIORI algorithm. Instead of building a new tree for each round, this procedure efficiently constructs a new set of candidates by merely copying nodes. Furthermore, during the counting phase, it passes through the tree and checks for the existence of candidates in the current transaction. This in contrast to the original algorithm, where for a given subset in the transaction, a search in a hash node is performed to check whether there is a candidate to be counted. It will appear that these both characteristics make this variant of APRIORI suitable for our purposes.

3 WARMR

As a first order extension of item sets, Dehaspe and De Raedt [1997] use sets of atoms, which they also refer to as *queries* when nearly all variables are existentially quantified and the set is ordered. The free variables are bound by a special purpose *key* predicate. The relation of the key and the query is illustrated in the following Horn clause:

$$\underbrace{k(X)}_{\text{key}} \leftarrow \underbrace{buys(X, Y, cardbonus), property(X, loyal)}_{\text{query}} \quad (2)$$

In this example the predicate $buys$ can be thought of as a table which describes the products that clients are buying, while the $property$ predicate refers to a table containing properties of clients. In the sequel we will use abbreviations such as b for $buys$, p for $property$ and c for $cardbonus$. This example shows how several tables can be combined more elegantly in a query than in an item set.

The support of the query is formalized using the key and is defined to be the number of variable bindings for which the key predicate can be proved. In the given example the support of $\{b(X, Y, c), p(X, l)\}$ is the number of variable bindings of X for which $k(X)$ can be proved given the Horn clause in Formula (2) and a knowledge base defined in PROLOG.

While for item sets the definition of the search space is straightforward, this is not the case for atom sets. Apart from the choice of predicate, there are also many possibilities for the usage of variables in the query. To define the *bias* of the search space WARMR uses a refinement operator based on *mode declarations*. Every mode declaration prescribes the way in which a predicate can be added to a query. The following is an example of a mode:

$$b(+, -, c). \quad (3)$$

It states that predicate b may be added to a query when the first parameter is bound to an existing variable, the second parameter introduces a new variable and the last parameter is bound to the constant c. The parameters are called *mode constraints*; here, we will call $+$ parameters and constant parameters *input parameters* and $-$ parameters *output parameters*. Often an integer is associated with every mode to indicate how many times at most the mode may be applied in the same query.

The usage of atoms instead of items turns it more difficult to create an efficient APRIORI-like algorithm: it is no longer reasonable to use the subset relation to express relations between atom sets. As replacement for the subset relation, and as approximation of logical implication, WARMR uses $\theta-$*subsumption*. An atom set C subsumes an atom set D, denoted by $C \succeq D$, if there is a substitution θ such that $C\theta \subseteq D$. The θ-subsumption relation induces an equivalence relation $\sim$, that is defined as follows: $C \sim D$ iff $C \succeq D$ and $D \succeq C$. It can be shown that a property similar to Formula (1) also holds for θ-subsumption on atom sets:

$$I_1 \succeq I_2 \Rightarrow support(I_1) \geq support(I_2). \quad (4)$$

For a set of frequent queries of size k (denoted by L_k) and a set of infrequent queries of size k (denoted by I_k), WARMR uses this algorithm to generate a new set of candidate queries C_{k+1}:

warmr-gen
$C_{k+1} = \emptyset$;
for all $c \in L_k$ **do**
  for all refinements c' of c **do**
    Add c' to C_{k+1} unless:
      (a) there is a $e \in \bigcup_{i \leq k} I_i : e \succeq c'$, or
      (b) there is a $e \in \bigcup_{i \leq k} L_i \cup C_{k+1} : e \sim c'$.

Restriction (a) removes queries which are *apriori* determined to be infrequent. Restriction (b) removes queries which have the same meaning as previously considered frequent queries or candidates. We will illustrate this on a small example. Consider the following set of mode declarations:

$$\{b(+, juice, -), t(+, c), t(+, electronicpurse)\}$$

This may lead to these two queries:

$$b(X, j, Y_1), t(Y_1, c), b(X, c, Y_2), t(Y_2, e)$$
$$b(X, j, Y_1), t(Y_1, e), b(X, c, Y_2), t(Y_2, c)$$

These clauses however have the same meaning and logically imply eachother.

A major problems of WARMR is that it heavily depends on a good implementation of $\theta-$subsumption. This is prohibitive as θ-subsumption is an NP-complete problem [Kietz and Lübbe, 1994].

4 FARMER

We propose two modifications of WARMR in order to make this algorithm more efficient. Each of the following two subsections will discuss one of them.

4.1 Knowledge base

When taking a closer look at the mode declarations, it can easily be seen that they can be mapped to procedures. As an example, consider the following facts: $b(1, ananas, c)$, $b(1, juice, e)$. Given mode $b(+, \#, +)$, this mode could be associated with a procedure which returns $true$ for $(1, j, e)$ and returns $false$ for $(1, a, c)$. Furthermore, a mode $b(+, -, -)$ could be associated with a procedure which for input 1 returns $\{(a, c), (j, e)\}$. In FARMER this idea is incorporated by binding all mode declarations to a procedure of one of these two types:

- boolean procedures, which for an input vector return true or false;
- outputting procedures, which for an input vector return a set of output vectors. Of course, this set may be empty and need not be computed entirely before all elements are used.

The first kind of procedures should be used in modes which do not have output parameters. The second kind is used in outputting modes.

A data structure for a knowledge base of PROLOG facts is created and accessed by procedures, as follows: for every mode declaration a multidimensional matrix is allocated; every element in the matrix corresponds to a set of input values and contains a truth value or a list of output values. When a fact for a predicate is read, all corresponding mode matrices are updated accordingly. The advantage of this mechanism is that it takes a constant amount of time to determine the truth of an atom, especially in our current implementation which stores the matrices in core memory[1]. Knowledge can however only be specified using facts or ad-hoc procedures.

4.2 Search

The search which FARMER performs differs in two aspects from the original WARMR algorithm:

- it does not use θ-subsumption;
- it manipulates a trie datastructure to generate the queries which are defined by the bias.

The trie datastructure is a tree which contains all candidate queries as a path from the root to a leaf. An example of such a trie is given in Figure 2. The tree is used for both counting and generating candidates.

[1]For large datasets, this could be a disadvantage. However, as our algorithm fits within the learning from interpretations approach, similar arguments hold when part of a database is on disk.

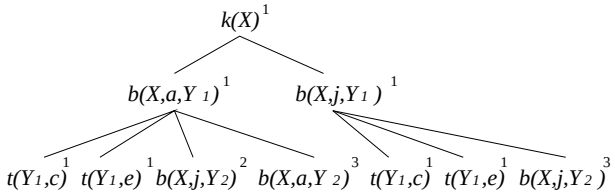


Figure 2: A trie in FARMER

Counting

We will describe the algorithm by giving pseudo-code. In this pseudo-code, we use the following notation:

- a capital A refers to an atom in the tree;
- a capital B refers to a set of variable assignments, which is a set containing a mapping from variables to values;
- $I(A, B)$ is an interpretation function, which returns true if an atom A can be proved using assignment B. In this function, one of the aforementioned procedures is used;
- $\mathcal{B}(A, B)$ is an assignment function. The assignment set B gives a value to some variables occuring in A. For the remaining unbound variables, this function returns all sets of possible assignments. The aforementioned outputting procedure is used here.

For all values of the key variables, the tree is traversed recursively, as follows:

```
Count(A,B)
if I(A,B) = true then
  if A is a leaf then
    disable A
    increase support A
    return true
  else
    for all B' ∈ B(A,B) do
      d := true
      for all A' ∈ children do
        if A' not disabled then
          d := Count(A',B ∪ B') ∧ d
        if d = true then
          disable A
          return true
return false
```

Initially all nodes are enabled. For a given node all values for the outputs are checked as long as there are children which have not been satisfied. A similar idea is also applied in [Blockeel *et al.*, 2000]. We show the integration of the procedures and mode declarations here.

Candidate generation algorithm

We will first introduce the mechanism of candidate generation by giving the algorithm. Afterwards we will compare this generation mechanism with the θ-subsumption based method of WARMR.

The generation mechanism is based on the following idea: when an atom with an input variable is moved to the beginning of a query, that variable could become an output variable, hereby violating the mode declarations. However, for

every atom in a query there is one first position at which it can occur without violation. All atoms that can be added to the end of a query can be subdivided in the following three classes:

1. atoms that could not have been added at an earlier position, as they use at least one new variable of the last atom in the query; we call these *dependent atoms*;
3. atoms that are a copy of the last atom in the query, except for the names of the output variables;
2. other atoms that could have been added at an earlier position.

Examples of these classes are given by the superscripts in Figure 2.

During the construction of the tree this subdivision is used. Given a trie and an ordered set of mode declarations, the trie is expanded as follows:

Expand(A)
if A is internal **then**
  for all $A' \in$ children **do**
    Expand (A')
else if A is frequent **then**
  add as child from left to right:
    1. all dependent atoms of A
    2. all frequent right brothers of A
    3. a copy of A with new output variables, if allowed
else
  remove A

The tree in Figure 2 is obtained using this mechanism when it is assumed that all queries of the following (typed) bias are frequent:

$$\{b(+A, a, -B), p(+A, j, -B), t(+B, c), t(+B, e)\} \quad (5)$$

The superscripts also in this case denote the mechanism that was used to create a node.

The first mechanism serves the purpose of introducing atoms which could not be added previously. The atoms are introduced in the same order as the corresponding mode declarations and a deterministic mechanism is used to go through all the input variables.

The dependent atoms are brothers of eachother; the second mechanism takes care that all subsets are generated afterwards – if not infrequent. By keeping the children in order, every subset is generated only once, or, equivalently, only one permutation out of a set of dependent atoms is considered. If necessary, the second mechanism gives new names to output variables to make sure they remain outputs.

The third mechanism is intentionally separated from the other two. Generation of repeating nodes is not desirable in many situations and should in any case be bound to a maximum. In our settings, the bias should explicitely state whether duplication of an atom is allowed.

Atoms of the third kind do not fit very well in the distinction that was introduced. A repeating node could in any case be exchanged with its parent. It would however not be efficient to introduce a set of identical nodes to overcome this problem. Later on, we will also see some additional disadvantages of these atoms.

Candidate generation discussion

Due to the absence of θ-subsumption, it can easily be seen that FARMER does not prune as many queries as WARMR does. In this section we will show which restrictions should be applied to the bias in order to make sure that FARMER will generate the same output.

In WARMR θ-subsumption is used for two purposes:

- to prune infrequent queries before counting;
- to remove queries which "mean the same" as other queries.

Only the θ-subsumption relation that is used for the latter purpose will be considered here, as only this relation influences the set of queries that is found. Infrequent queries will not occur in the results even if they are not pruned.

The θ-subsumption equivalence relation is only one method for determining that queries mean the same. A less strict relation is the equality relation under substitution, which we will denote with $\simeq$ here and is defined for two (unordered) sets of atoms as follows: $C \simeq D$ iff there exist substitutions θ_1 and θ_2 such that $C\theta_1 = D$ and $D\theta_2 = C$. The correspondence between these relations can be expressed using Plotkin's reduced clauses.

Definition 4.1 *A clause D is called reduced iff $C \subseteq D$ and $C \sim D$ imply $C = D$. [Plotkin, 1969]*

Theorem 4.1 *Let C and D be reduced sets of atoms. Then $C \simeq D$ iff $C \sim D$.*

Proof "⇒": this is clear as $C\theta_1 \subseteq D$ and $D\theta_2 \subseteq C$.
"⇐": as $C \succeq D$, $C\theta_1 \subseteq D$, and as $D \succeq C$, $C\theta_1\theta_2 \subseteq C$. Let $C' = C\theta_1\theta_2$. Because $C' \subseteq C$ and $C\theta_1\theta_2 \subseteq C'$, also $C \sim C'$ holds (by definition), and then $C = C'$ because C is reduced. Thus $C\theta_1\theta_2 = C$. In the same way, $D\theta_1'\theta_2' = D$. As $|C\theta_1| = |C|$ and $C\theta_1 \subseteq D$, $|D| \geq |C|$. As $|D\theta_1'| = |D|$, $|C| \geq |D|$, and finally $|D| = |C|$. By combining $|C\theta_1| = |D|$ and $C\theta_1 \subseteq D$, $C\theta_1 = D$ is shown. This proves that $D \simeq C$. □

From this theorem it also follows that if atom sets are reduced, they can never be subsumption equivalent when they differ in length.

We will show that for a restricted bias, FARMER will always generate reduced atom sets. Then we will show that FARMER does not generate two different atom sets that are substitution equivalent. From this we conclude that, given a restricted bias, FARMER will not generate queries that subsume eachother.

Definition 4.2 *A redundancy restricted bias should obey the following rules:*

1. *no functions may be used;*
2. *repetition of an atom by an attom which differs only in the name of the output variables is not allowed;*
3. *no two modes for the same predicate may exist for which the constraint parameters differ, unless the corresponding parameters are both constant parameters.*

The second rule prevents queries such as $b(A, a, B_1)$, $b(A, a, B_2)$ from being generated. The third rule disallows the biases $(b(+, a, -),\ b(+, -, -))$ and $(b(+, a, -),$

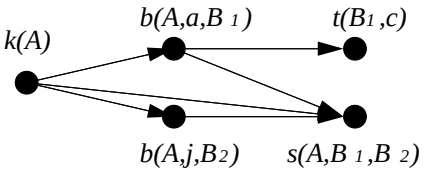


Figure 3: A partial order for a query

$b(+,a,+)$, $t(+,-)$), and consequently queries $(b(A,a,B_1)$, $b(A,C_1,B_2))$ and $(b(A,a,B_1)$, $t(B_1,B_2)$, $b(A,a,B_2)))$. Query $b(A,a,B_1), t(B_1,c), b(A,j,B_2)$ remains possible.

Theorem 4.2 *For a redundancy restricted bias* FARMER *will always generate reduced atom sets.*

Proof Assume that D is a query in the trie, obtained by using a redundancy restricted bias. We will show that for any subset $C \subset D$, $D\theta = C$ can never be true. In order for this, there must at least be two atoms A_1 and A_2 in D which are mapped to the same atom in C: $A_1\theta = A_2\theta$. For any pair we will try to construct such a substitution. By definition of the bias, both atoms must have inputs at the same positions (restriction 2), while the input variables must be different (restriction 1 in combination with the tree building procedure, where such atoms could only be generated as brothers). Construct a substitution which unifies A_1 and A_2. This substitution will always map variables to variables, as no functions are allowed and no modes with constants and variables at the same parameters. Apply θ to the whole query. Consider the set of atoms that introduced the variables used in A_1 and A_2, then there are two possibilities:

1. This set contains one atom which has two outputting parameters. By θ these are bound to the same variable. Such an atom can never be generated according to the mode mechanism used by FARMER;
2. This set has at least two different atoms. Of both atoms an output is bound to the same variable by θ. In whatever order these atoms are placed, one of them has now an input at a position where an output occured. This would require another mode, which is not allowed in this restricted bias.

Thus there can not exist redundant queries. □

Theorem 4.3 *Given a redundancy restricted bias,* FARMER *will never generate two queries that are substitution equivalent.*

Proof We first remark that for ordered atom sets, such as queries, a deterministic variable numbering can be used. Furthermore we note that two queries must be of equal size and that the substition can only map from variables to variables. Thus, to determine whether two queries substitution equal eachother, it suffices to find a permutation of atoms, followed by a variable renumbering, that makes two queries equal. We will show that FARMER generates one permutation.

The restricted bias is such that for every atom in an (unordered) atom set, there is only one possible mode declaration. The usage of input and output parameters determines a partial order on the atoms, which can be depicted in a graph such as in Figure 3 for the atom set $\{b(A,a,B_1), b(A,j,B_2), t(B_1,c), s(A,B_1,B_2)\}$ and the bias $(b(+A,a,-B), b(+A,j,-B), t(+B,c), s(+A,+B,+B))$. Use this strategy to order the nodes in a query Q:

order(A)
add A to the end of Q
S := nodes with incoming arrow from A and
  no incoming arrow from outside Q
order S according to mode declarations and
  a deterministic input variable numbering strategy
for all $A' \in S$ in order **do**
  order(A')

The order obtained by this strategy corresponds to the order of FARMER: the set S corresponds to the set of dependent nodes; the tree building mechanism which places new nodes before copied nodes takes care of the recursion by acting as a sort of LIFO queue. □

Corollary 4.1 *Given a redundancy restricted bias* FARMER *will never generate queries that θ-subsume eachother.*

5 Experimental results

We have compared FARMER and WARMR on two datasets. We should remark that in our experiments we used an implementation of WARMR that did not yet use the tree datastructure discussed in [Blockeel *et al.*, 2000]; a comparison of both algorithms is therefore not completely fair. Experiments in [Blockeel *et al.*, 2000] revealed speed-ups of 20 for WARMR in some situations.

Bongard

The Bongard dataset [Bongard, 1970] contains descriptions of artificial images. The task is to discover patterns in the images. Every image consists of several figures that can be included into eachother. No redundancy restricted bias can be used.

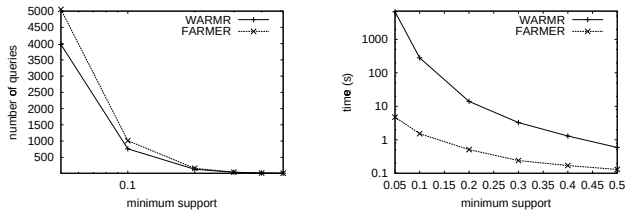
In Figure 4 the results of the experiments are depicted. Figure 4(a) shows the number of queries that each algorithm finds. The number of FARMER is higher in all cases, which is also expected for this bias. In Figure 4(b) the execution times of the algorithms are compared[2]. Paying attention to the fact that the scale is logarithmic, the speed-ups are considerable for this dataset.

Frequent itemsets

In this experiment we compare the performance of FARMER to a special purpose algorithm. As test case a binary digit dataset is used which contains 1000 binary coded numbers. The special purpose algorithm which is used as comparison is the breadth-first implementation of APRIORI by [Pijls and Bioch, 1999]. FARMER uses many of the mechanisms introduced in that algorithm and should perform comparable to that algorithm.

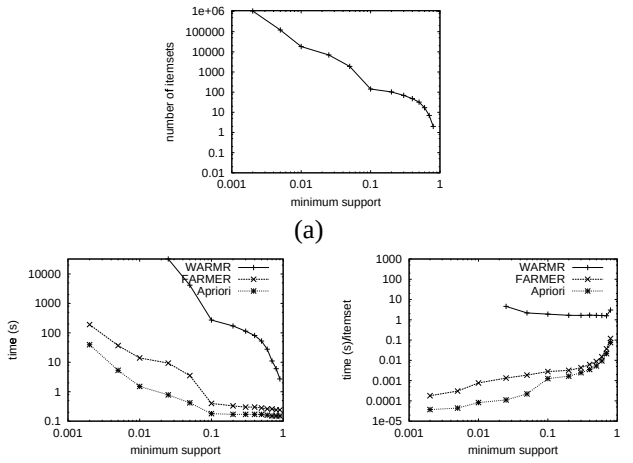
In Figure 5 the results of the experiments are depicted. A characteristic of the dataset is given in Figure 5(a). The number of frequent itemsets appears to increase rapidly when the

[2]Experiments were carried out on a Sun Enterprise 450 4x400MHz UltraSPARC2 CPU w/4MB E-cache 4GB RAM.



(a) (b)

Figure 4: A comparison of FARMER and WARMR on the Bongard dataset. (a) The number of queries in the output. (b) Execution times in seconds. Note that the scales are different.



(a) (b) (c)

Figure 5: Comparison of results for the digit dataset. (a) The number of frequent itemsets for each minimum support. (b) The execution times of the algorithms. (c) The execution times consumed for each itemset. Note that the scale is logarithmic on both axis.

minimum support is beneath 0.1. In Figure 5(b) the execution times of the algorithms are given. The time graph of FARMER is comparable to that of APRIORI, although still exponentially larger. WARMR has a completely different behaviour than both other algorithms and had such high execution times that no experiments were carried out for low supports.

In Figure 5(c) both previous graphs are combined and the execution time for each itemset is shown. It makes clear how the algorithms react when the amount of solutions they have to find increases. While the execution times of WARMR increase, the times of the other algorithms decrease. Although the overhead for each itemset is larger in FARMER, which could be explained by the additional mechanisms that are hooked in, the difference is acceptable. The decreasing trend can be explained by the increasing number of overlapping evaluations when the number of itemsets increases.

6 Conclusions and further work

We introduced an efficient algorithm for discovering queries. It uses a tree datastructure both to count queries as to generate queries. We showed that for a restricted type of bias, this algorithm is equivalent to a previous algorithm, WARMR, and performs much better.

Although we believe that our restricted bias already adds considerable expressive power to propositional association rules, we are looking at some possibilities to overcome these restrictions. It appears that in case the second restriction is lifted, the range of possible rules already increases considerably. We are investigating the possibility of using the order of the tree in combination with a more sophisticated default order of queries.

Furthermore, we plan to perform more experiments. We successfully performed some experiments on a database with one million records, but more experiments are necessary to find out the behaviour of FARMER on datasets of this size.

References

[Agrawal *et al.*, 1996] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining.*, pages 307–328. AAAI/MIT Press, 1996.

[Blockeel *et al.*, 2000] H. Blockeel, L. Dehaspe, B. Demoen, G. Janssens, J. Ramon, and H. Vandecasteele. Executing query packs in ilp proceedings of ilp2000 - 10th international conference on inductive logic programming, 2000.

[Bongard, 1970] M. Bongard. *Pattern Recognition*. Hayden Book Company (Spartan Books), 1970.

[Dehaspe and De Raedt, 1997] L. Dehaspe and L. De Raedt. Mining association rules in multiple relations. In S. Džeroski and N. Lavrač, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297, pages 125–132. Springer-Verlag, 1997.

[Dehaspe *et al.*, 1998] L. Dehaspe, H. Toivonen, and R. D. King. Finding frequent substructures in chemical compounds. In R. Agrawal, P. Stolorz, and G. Piatetsky-Shapiro, editors, *4th International Conference on Knowledge Discovery and Data Mining*, pages 30–36. AAAI Press., 1998.

[Kietz and Lübbe, 1994] J-U. Kietz and M. Lübbe. An efficient subsumption algorithm for inductive logic programming. In S. Wrobel, editor, *Proceedings of the 4th International Workshop on Inductive Logic Programming*, volume 237, pages 97–106. Gesellschaft für Mathematik und Datenverarbeitung MBH, 1994.

[Nijssen, 2000] S. Nijssen. Data mining using logic, master's thesis, Leiden University, 2000.

[Pijls and Bioch, 1999] W. Pijls and J. C. Bioch. Mining frequent itemsets in memory-resident databases. In E. Postma, editor, *Proceedings Eleventh Belgium/Netherlands Artificial Intelligence Conference*, pages 75–82, 1999.

[Plotkin, 1969] G. D. Plotkin. A note on inductive generalization. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pages 153–163, Edinburgh, 1969. Edinburgh University Press.

A Simple Feature Selection Method for Text Classification

Pascal Soucy and Guy W. Mineau
Dept. of Computer Science
Université Laval, Québec, Canada, G1K 7P4
{Pascal.Soucy, Guy.Mineau}@ift.ulaval.ca

Abstract

In text classification most techniques use *bag-of-words* to represent documents. The main problem is to identify what words are best suited to classify the documents in such a way as to discriminate between them. Feature selection techniques are then needed to identify these words. The feature selection method presented in this paper is rather simple and computationally efficient. It combines a well known feature selection criterion, the information gain, and a new algorithm that selects and adds a feature to a bag-of-words if it does not occur too often with the features already in a small set composed of the best features selected so far for their high information gain. In brief, it tries to avoid considering features whose discrimination capability is sufficiently covered by already selected features, reducing in size the set of the features used to characterize the document set. This paper presents this feature selection method and its results, and how we have predetermined some of its parameters through experimentation.

1 Introduction

In automatic text classification most techniques use *bag-of-words* to represent documents. The main problem is then to identify what words are best suited to classify the documents in predefined classes. Feature selection techniques are then needed to identify these words.

Many feature selection techniques use some metric to determine the relevancy of a term with regard to a classification criterion. One such technique uses the information gain metric assessed over the set of all words encountered in all texts, and then chooses the best k words according to that metric [Lewis and Ringuette, 1994]. Because of computational efficiency issues, it may not be possible to use the learning algorithm itself to determine which k is best suited to learn the descriptions of the classes of documents so that the accuracy rate of the algorithm is maximized. In that case, there is little guidance to determine what value k should be. Some naive techniques just predetermine a k using the learning algorithms over a small set of data sets, and then uses that value of k directly on other data sets even though these data sets may be quite different from those used to learn the value of k. To our opinion, this k becomes rather arbitrary unless data sets can be characterized in such a way to be easily associated with previously tested data sets.

The feature selection method presented in this paper is rather simple and computationally efficient. Efficiency is a major issue in text classification of large corpora (like found on the WWW) since many thousand features are usually involved. Our method is a filter approach [Hall and Smith, 1999], as the learning algorithm is not involved in the selection process of features and is self-adaptable to the nature of the corpus. It combines a feature selection method known to be among the best for text classification [Yang and Pedersen, 1997], that is, thresholding on information gain, and a new algorithm that selects and add features to a bag-of-words according to their average coocurrence in the training set with the features already in a small set composed of the best features selected so far for their high information gain, called the pool. This paper presents this feature selection method and how we have predetermined some of its parameters, called thresholds, through experimentation. It also shows how these thresholds can be used as such in other domains and still render a good performance level in terms of the relevancy of the feature set extracted for subsequent classification purposes.

2 Experimental Testbeds

2.1 Learning Algorithms

In order to assess the potential of our feature selection method, we used it with two different learning algorithms that are widely used in text classification and retrieval: k-nearest neighbors (KNN) algorithms (with weighted and non weighted features according to different weighting functions) and Rainbow, a naive Bayes classification algorithm for text classification [McCallum, 1996; Fürnkranz *et al*., 1998]. However, because of lack of space, this paper only presents the results obtained with the non weighted features KNN algorithm and Rainbow, since all weighted KNN algorithms performed more poorly than their non-

Task name	Dataset	Classes	Size[1]	Train/Test Balance
WebKBCourse	WebKB[2]	Course/noncourse	400	80/320
ReutersCornWeat	Reuters-21578[3]	Corn/wheat	400	40/360
Ling-Spam	Ling-Spam[4]	Spam/non-spam	120	40/80
WWWPrisoner	WWW[5]	Relevant/irrelevant	60	40/20
WWWBeethoven	WWW[6]	Relevant/irrelevant	60	40/20
UsenetNews1	Usenet[7]	Soc.history/ Soc.religion.christians	67	40/27

Table 1 : The data sets used in our experiments. Each data set is a binary classification task between two classes. Size is the total number of documents in the data set. Train/Test Balance is the number of documents in the training and testing sets respectively. Note that the training sets have been voluntarily kept small for we believe that for most applications, few examples are available or if larger sets are available, manual labeling of the documents becomes prohibitive.

weighted counter-part. Appendice A briefly describes the weighting functions tested to try and improve the non-weighted version of our KNN algorithm, without success.

2.2 Data Sets

For our experiments, we used data sets previously studied in other work, and our own data sets. The data sets used in our experiments are described in table 1. These data sets may be categorized in two types: those including documents written with a concise vocabulary and those more freely written. The former set includes data sets such as Reuters (brief news) and WebKB (Web pages mainly retrieved from computer science university departments) while the latter set includes noisy data sets such as Ling-Spam (consisting of emails belonging to a mailing-list, including junk emails), Prisoner (the TV show) and Beethoven (the composer) (Web pages retrieved with Altavista using a keyword search), and News (unmoderated Usenet newsgroups on history and christianism). The difficulty associated with a classification task depends highly on the type of vocabulary used [Scott and Matwin, 1999]. Words were stemmed using the Porter algorithm for our KNN algorithms, and using the –use-stemming option with Rainbow.

[1] The size may be different from the original data set to ensure computationally manageable experiments. In that case, subsets of the original data sets have been selected randomly. For WWW pages research, the size is small due to the fact that there are very few relevant documents.

[2] Described in [8] and is available at www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/.

[3] Well known data sets available at http://www.research.att.com/~lewis/reuters21578.html.

[4] The Ling-Spam anti-spam filtering corpus, available at: http://www.iit.demokritos.gr/~ionandr/lingspam_public.tar.gz.

[5] WWW pages manually classified according to their relevance to a subject, the TV show *The Prisoner*.

[6] WWW pages manually classified according to their relevance to a subject, Beethoven biographies.

[7] Usenet articles collected from soc.history and soc.religion.christians newsgroups. Only the first message of a thread of discussion has been retrieved. See [Scott and Matwin, 1999] for explanations about this choice.

3 Feature selection

Information Gain (IG) in text classification measures the reduction in entropy (heterogeneity of a set) gained using a term presence or absence to determine class membership. It is often used in text classification in the bag-of-words approach [Joachims, 1998; Nigam *et al.*, 1999; Sahami *et al.*, 1998]. This measure is the same used by ID3, a decision tree learning algorithm, to select the best attribute at each step of the tree construction [Quinlan, 1986].

3.1 The *k* best features using the Information Gain

Under this approach, a text classification task is considered using the first *k* words that rank highest on that metric. That number *k*, however, depends highly on the kind of corpus and task involved. Under a Bayesian classification approach, many noticed that a naive Bayesian classifier works best with many features, but better if not every available feature is used. From various experiments it was noticed that the *k* which seemed to be an approximation of the average number of features leading to reasonable results with that algorithm varies from 2000 features [Craven *et al.*, 1998] to 500 features [Sahami *et al.*, 1998] depending on the nature of the corpus.

These findings show a great variation in the size of the required vocabulary. If we consider that Bayesian classifiers suffer when irrelevant features are kept as selected features, it is difficult to justify why a predefined number of features would allow to optimize the accuracy rate of the classifier. Actually, in such data sets as Reuters, there are some words that are so characteristic of some class that fewer features need to be selected than with a data set involving newsgroups or email messages. It seems obvious that the number of required features highly depends on the nature of the corpus and of the learning algorithm. Therefore, how could we characterize the nature of the corpus in terms of the available features so that guidelines on how to select a proper *k* are easily inferred? This paper proposes one such way to determine *k* according to the nature of the terms composing the vocabulary with regard to their discrimination power relative to one another.

3.2 Using Information Gain with a threshold

A simple feature selection method could look at the IG of each feature (taken individually) with regard to the classification task. For instance, there are methods that select any feature with an IG greater than some threshold [Yang and Perdersen, 1997]. This is fully relevant to Bayesian classifiers. It is easy to see that features with low IG are useful to characterize only a handful of documents. Therefore, their probability distribution over the whole set of documents can not be estimated correctly. Bayesian classifiers rely on these probability estimates.

Similarly, infrequently used dimensions (attributes) in a KNN approach introduce noise more than anything else. As a matter of fact, KNN seems to produce better results when fewer features are involved, using the IG to keep the best features [Joachims, 1998]. KNN bias depends mostly on the kind of similarity function it uses, but it always uses the vector model, a weakness of this model being the fact that features are considered to be independent of each other. The function that seemed most successful (see Appendix A, a non weighted cosine similarity function), assigns to each feature the same weight. Given that, we should only consider features that are promising in terms of class characterization, as the presence of many irrelevant features diminish the influence of very good (relevant) features. Choosing a fixed number of features may lead to consider irrelevant ones (as with the Reuters data sets) if k is too high, or remove some good features (as with junk email classification) if k is too low.

However, even if we obtained good results on average using the KNN and Bayes classifiers, by looking at each data set individually we noticed that it would be possible to restrict the set of selected features even more. For instance, with the Reuters data set, we saw that in most binary classification tasks (for example, classifying documents according to their relevance to "corn" or "wheat"), only a few words (often under 10) were needed to obtain the best accuracy score. We observed that high IG features could represent classification alternatives (in which case one is useless), or could be complementary. Therefore, we could assess the cooccurrence of highly ranked features and eliminate those that represent alternatives of already selected high IG features.

We conducted extensive experiments where we had the number of features ranging from 1 to 5000 and the IG threshold ranging from 0 to 1. From all these experiments, we found that the best threshold to classify the data sets that we had was 0.1 in average. Therefore, we propose to first identify the terms whose IG is higher or equal to 0.1, and then, select only those features that are not covered by the strong features already in the set (in terms of their discrimination capability), as explained below. In what follows, we prove that this further refinement of the term set improves both the accuracy of the classifier and the interpretability of the characterization so produced since the number of selected features is sensibly reduced.

3.3 Our Feature Selection Algorithm

The algorithm of Figure 1 proposes a refinement of the set of selected features (already offering an IG higher than some threshold) by removing those features that cooccur with very high IG features. Cooccurrence is a kind of measure of dependency between features, which is a weakness of most statistical models used today to characterize a set of features. Because of computational issues, dependencies between terms are usually ignored. We propose to identify a small subset of terms that we know are relevant for the classification task at hand (according to their individual IG), and to compute dependencies between any new term and these preselected terms. The cooccurrence metric that we consider

A Simple Feature Selection Algorithm: mCooccurrence

```
Let V = set containing every word found in the corpus,
        sorted by Information Gain (in decreasing order)
Let Pool = { } (the pool)
Let D = set of documents
Let igain_thres = 0.1
Let cooc_thres = 0.45
Let nbkeep = 5

Pool ← nbkeep first features from V

FOR each word w IN V - Pool
  IF IG(w,D) < igain_thres THEN
      V = V – {w}
  ELSIF μCooc(w,Pool,D) > cooc_thres
        THEN V = V – {w}
  ENDIF
ENDFOR

μCooc (w, Pool,D) :
  Let Sum = 0
  FOR each p_i in Pool
      Sum ← Sum + (P(w | p_i) in D)
  RETURN (Sum / nbkeep)
```

Figure 1 : A Simple Feature Selection Algorithm

in the algorithm of Figure 1, written $\mu Cooc(w,Pool,D)$, is based on a partial cooccurence probability of w with each word in $Pool$, considering the whole training set. It is computed as the mean value of $P(w|p_i)$ for all p_i in $Pool$, where $P(w|p_i)$ is the conditional probability to find w in any document if it already contains p_i.

Let us take an example and assume that the word "corn" has an IG value of 0.95, meaning that it almost perfectly divides the training set. Then let us suppose that P(starch|corn) = 1 and IG(starch) < IG(corn). Since "starch" always occurs when "corn" does and since its IG is lower

than that of "corn", "starch" is unlikely to bring new information (in terms of discrimination power) to the feature set, and could therefore be removed from it.

Needless to say that the reality is not that perfect. Removing a feature only when it is *totally* covered by some other feature (in terms of its discrimination capability) would remove only a few features, if any, which goes against our aggressive selection objective. Some approximation of coverage between terms is thus required to effectively reduce the vocabulary to those features that are meaningful in terms of their overall discrimination power. Our approach is to consider a mean of partial cooccurrence between any new term and the terms in P, the set of the *nbkeep* highest ranked features according to their IG, called the pool. Of course, *nbkeep* and *cooc_thres* (see the algorithm of Figure 1) have been empirically determined using our own data sets: Prisoner, Beethoven and News, and tests have been conducted over previously studied data sets: *Reuters*, *Ling-Spam* and *WebKB*, in order to prevent over fitting the parameters on these data sets. Due to a lack of space, we won't show here the results of this parameterization, but we concluded from these tests that the values used in the algorithm of Figure 1 hold for all our data sets, and seem to hold for other data sets that we are currently experiencing with.

4 Results

Table 2 shows the results obtained with the different feature selection methods that we tested over each corpus, under our two learning algorithms: a non weighted features KNN algorithm (first three columns) and Rainbow, a well-used Bayesian classifier (last five columns). For each method, we show *k*, the number of features selected and used thereafter by the learning algorithm, and the accuracy score (%) obtained subsequently.

The first column shows the *k* features selected when we set a threshold of 0.1 on the Information Gain score (IG) of each feature. Again, this threshold was assessed empirically over all possible values of IG ranging from 0 to 1. From the resulting set, our µCooccurrence refinement algorithm was applied to produce a smaller set of features, and the results are shown in the second column. One can see that the number of features is almost always reduced, and if so, significantly and without impact on the accuracy rate, meaning that we could eliminate useless features in terms of the prediction capability of the learning algorithms used. This helps produce a characterization of the learned concepts with a smaller vocabulary, and therefore facilitates the interpretation of the results and reduces the computation cost to classify new documents.

Then we wondered if this minimal number of features could have been reached through a fine tuning of the IG score that we used. We found that an IG of 0.13 would produce the same number of selected features (taken over all corpora), but that the accuracy rate would then go down, as shown in the third column. Obviously, the selected features in this case are not the same as those of the previous experiment. Furthermore, we see that the small fluctuation in the IG rate (0.03) has great impact on the individual corpora in terms of the number of selected features. Considering our detailed search for a good IG rate, it is not surprising to see such a fluctuation and its negative impact on the accuracy rate of the classifiers. This tends to show that 0.1 is a better threshold than 0.13, as validated in a previous experiment, and that our feature selection method is therefore useful to reduce that set by removing useless (or not so useful) features. It also suggests, at least on this type of corpora, that our method is sound and useful to determine a relevant *k* which, without such a method, would be impossible to determine.

For the Bayesian classifier, we observe the same phenomenon. The forth column gives the number of selected features for IG = 0.1 and the accuracy rate obtained after training Rainbow with these features. For comparison purposes, we also trained Rainbow with 2000 features (fifth column) as is suggested in [Craven *et al.*, 1998]. By comparing the forth and fifth columns we can see that the 2000 feature sets performs better than what we obtain with IG = 0.1 except in cases where there are a few words with extreme discrimination capabilities like with the Reuters and Beethoven corpora. Despite this small loss in accuracy, the feature set with IG = 0.1 is nevertheless significantly smaller than 2000, which would undoubtedly help the interpretation of the learned concepts.

In the next experiment (sixth column), we applied our µCooccurrence refinement algorithm to reduce even further the size of the feature set. Again we obtain a reduced set without any impact on the accuracy rate, facilitating even more the subsequent interpretation of the learned concepts.

However, by setting IG = 0.13, we obtain the same number of features (taken over all feature sets) and a slightly smaller accuracy rate (see the seventh column).

Finally, as expected, an unbounded number of features has a negative impact on the accuracy rate of a Bayesian classifier, as shown in the last column. Though not illustrated in Table 2, this phenomenon also holds for KNN classifiers when the number of attributes is extremely high and the number of relevant attributes is much lower than the number of attributes, which is the case in text classification. Therefore, there is a need to determine a proper *k* for training the classifier with the relevant attributes.

5 Complexity Issues

Let *N* be the number of documents, let *V* be the size of the vocabulary, and let *P* be the size of the pool. By choosing to compute cooccurrence only between the elements of the vocabulary and the pool, we avoid having to compute it for all $O(V^2)$ possibilities (when pairs of features are considered), producing a $O(VPN)$ complexity factor rather than $O(V^2N)$ for assessing the cooccurrence of terms over all *N* documents. Since computing the IG for all features in *V* is

Data Set	IG = 0.1 KNN		µCooc KNN		IG = 0.13 KNN		IG = 0.1 Bayes		k=2000 Bayes	µCooc Bayes		IG = 0.13 Bayes		All features Bayes	
	k	**%**	**k**	**%**	**k**	**%**	**k**	**%**	%	**k**	**%**	**k**	**%**	**k**	**%**
WebKBCourse	37	97.5	35	96.9	24	95.3	37	95	95.6	35	95	24	94.1	12150	94.2
ReutersCornWheat	8	98.3	8	98.3	6	98.3	8	97.4	89.2	8	97.4	6	97.4	3393	81.1
Ling-Spam	86	95	77	95	47	91.3	86	86.3	90	77	86.3	47	87.5	4484	86.3
WWWPrisoner	53	85	9	90	50	90	53	70	80	9	70	50	70	5076	70
WWWBeethoven	121	85	8	85	106	80	121	90	85	8	90	106	85.7	3327	90
UsenetNews1	184	85.2	130	85.2	35	85.2	184	92.6	96.3	130	92.7	35	92.6	8522	92.6
Micro-average	**95.6**		**95.5**		**93.7**		**92.7**		**93.1**	**92.7**		**92.2**		**90.3**	
Number of features	**489**		**267**		**268**		**489**		**12000**	**267**		**268**		**36952**	

Table 2 : Number of features (k) and accuracy score (%) for each data set under different feature selection and learning methods. The total number of features is the summation of the *k* column; the micro-average calculates the average of correctly classified instances over all data sets and is therefore not an average of the individual accuracy scores.

O(*VN*), and since ordering them by their IG is O(*V* log *V*), for large corpora (where $\log_2 V << N$), the dominating complexity factor of our algorithm is the assessment of the cooccurrence of terms. Consequently, our method works under a complexity factor of O(*VPN*). Provided that we can determine *P* independently of *V* and *N*, as was done in our experiments, the resulting complexity figure for our method then becomes O(*VN*), which is the same complexity figure for initially assessing the IG of each feature in the vocabulary. Consequently, our µCooccurrence feature selection algorithm has the same asymptotic complexity as a feature selection method that solely computes the IG of each feature in its feature set, with a hidden constant that has now doubled however. Nevertheless, the method is computationally efficient.

6 Conclusion and Future Work

We have presented in this paper a very simple feature selection method for text classification that refines a set of features previously selected according to an IG criterion. This refinement is based on the cooccurrence of new features with a predetermined subset of highly ranked features, called the pool, hoping to avoid to include features whose discrimination capability with regard to the task is (partially) covered by the features of the pool. We seek to improve the classification speed of the learner without reducing accuracy and reduce the set of attributes that will be used later on to support the interpretation of the learned concepts. This paper shows that we reached this objective for the corpora that we studied. The method that we propose in this paper certainly proved to be promising for certain types of text classification tasks, as demonstrated by our experiments. At first glance, we are tempted to conclude that µCooccurrence has the capability to remove the features that brings no further information to those already selected. Of course further experiments will be conducted to assess the validity of that claim under various conditions.

Additional investigation will also be conducted in order to understand why µCooccurrence seems to have more impact on (non weighted feature) KNN algorithms than on Bayesian classifiers. Though we feel that the probabilities that ponder the involvement of attributes provides some explanation, we still can not understand why weighted features KNN approaches failed to surpass the non weighted KNN algorithm that we used, despite extensive experimentation using a wide range of distance functions. [Domingos and Pazzani, 1997] demonstrated why naive Bayes learners perform well in spite of the (so-called) word independence assumption. Therefore, we will focus our investigation towards the study of word dependence and its impact on the KNN model.

Further effort is also needed to extend the µCooccurrence algorithm so that it can handle multiple concept learning. Our experiments have been conducted only over binary classification tasks.

Another field of investigation will be to use multistrategic approaches to feature selection. For instance, we could assess the information gain of *sequences* of features. We could base our choice of feature sequences on the first few levels of decision trees built from our initial feature set, and increment the vocabulary accordingly.

Finally, though we set the IG and cooccurrence thresholds at 0.1 and 0.45 through extensive experimentation, more study on the impact of these parameters on the classification task with regard to different types of corpora is required.

Appendix A

The learning bias of the KNN model is mainly related to its distance (similarity) function and to its neighbor weighting function to compute that distance. Distance weighted KNN adjusts the importance of each neighbor by weighting their contribution to the classification of a new instance. In this section, we consider the weights associated with the features themselves rather than with individual instances.

This section presents the weight and distance functions that have been studied in our search to determine which KNN algorithm would be used in the assessment of our feature selection method. Surprisingly, the simplest of these functions worked better that the others.

Let us define an instance x is as

$$<a_1(x), a_2(x), a_3(x), \ldots, a_n(x)>$$

where $a_j(x)$ is the weight of the jth attribute (feature) on x.

Using non-weighted features

If we consider that the weight of each feature is the same, then we have:

$a_j(x) = 1$ If the word is in document x

$a_j(x) = 0$ otherwise

Using weighted features

The goal reached using weighted features is to stretch the neighborhood space so that irrelevant features are given less importance. There are several ways to do this feature weighting. Here are a few examples that were tested:

- the weight of a feature for a document is its IG if the word is present in the document, 0 otherwise
- the weight of a feature is its frequency in the document
- the weight of a feature is its TFIDF (*Term frequency/Inverse Document Frequency*)

We also tried many variants of these methods, for example, squaring the IG so that features with high IG would get yet higher weights. Under these various weight assignment policies, we experimented with two distance functions :

Euclidian distance function

$$d(x1, x2) = \sqrt{\sum_{j=1}^{n} \left(a_j(x1) - a_j(x2)\right)^2}$$

Cosine similarity function [Salton89]

$$\text{CosSim}(x1, x2) = \frac{\sum_{j=1}^{n} a_j(x1) \cdot a_j(x2)}{\sqrt{\sum_{j=1}^{n} a_j(x1)^2 \cdot \sum_{j=1}^{n} a_j(x2)^2}}$$

References

[Craven *et al*., 1998] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. *Technical report*. Departement of Computer Science. Carnegie Mellon Univ.. 1998.

[Domingos and Pazzani, 1997] P. Domingos and M. Pazzani. Beyond independence: Conditions of the optimality of the simple bayesian classifier. *Machine Learning*. (29) pp 103-130. 1997.

[Fürnkranz *et al*., 1998] J. Fürnkranz, T. Mitchell and E. Riloff. A Case Study in Using Linguistic Phrases for Text Categorization on the WWW. *Working Notes of the 1998 AAAI/ICML Workshop on Learning for Text Categorization*.

[Hall and Smith, 1999] M. A. Hall and L. A. Smith. Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper. *Proceedings of the Florida Artificial Intelligence Symposium* (FLAIRS-99). 1999.

[Joachims, 1998] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proceedings of the European Conference on Machine Learning*. Springer. 1998.

[Lewis and Ringuette, 1994] D. D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. *Third Annual Symposium on Document Analysis and Information Retrieval*. pp 81-93. Las Vegas. 1994.

[McCallum, 1996] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. *http://www.cs.cmu.edu/~mccallum/bow*. 1996.

[Nigam *et al*., 1999] K. Nigam, J. Lafferty and A. McCallum. Using Maximum Entropy for Text Classification. *IJCAI-99 Workshop on Machine Learning for Information Filtering*. 1999

[Quinlan, 1986] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1) pp 81-106. 1986.

[Sahami *et al*., 1998] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian approach to filtering junk e-mail., *AAAI/ICML Workshop on Learning for Text Categorization*. Wisconsin. 1998.

[Salton, 1989] G. Salton. *Automatic text processing: the transformation, analysis, and retrieval of Information by Computer*. Reading, MA: Addison-Wesley. 1989.

[Scott and Matwin, 1999] S. Scott and S. Matwin. Feature engineering for text classification. *Proceedings of ICML-99, 16th International Conference on Machine Learning*. pp. 379-388. Morgan Kaufmann Publishers. 1999.

[Yang and Perdersen, 1997] Y. Yang and J. P. Pedersen. A Comparative Study on Feature Selection in Text Categorization. *Proceedings of the Fourteenth International Conference on Machine Learning* (ICML'97). 1997.

Link Analysis, Eigenvectors and Stability

Andrew Y. Ng
Computer Science Division
U.C. Berkeley
Berkeley, CA 94720

Alice X. Zheng
Computer Science Division
U.C. Berkeley
Berkeley, CA 94720

Michael I. Jordan
CS Div. & Dept. of Statistics
U.C. Berkeley
Berkeley, CA 94720

Abstract

The HITS and the PageRank algorithms are eigenvector methods for identifying "authoritative" or "influential" articles, given hyperlink or citation information. That such algorithms should give consistent answers is surely a desideratum, and in this paper, we address the question of when they can be expected to give stable rankings under small perturbations to the hyperlink patterns. Using tools from matrix perturbation theory and Markov chain theory, we provide conditions under which these methods are stable, and give specific examples of instability when these conditions are violated. We also briefly describe a modification to HITS that improves its stability.

1 Introduction

Recent years have seen growing interest in algorithms for identifying "authoritative" or "influential" articles from webpage hyperlink structures or from other citation data. In particular, the HITS algorithm of Kleinberg [1998] and Google's PageRank algorithm [Brin and Page, 1998] have attracted the attention of many researchers (see also [Osareh, 1996] for earlier developments in the bibliometrics literature). Both of these algorithms use eigenvector calculations to assign "authority" weights to articles, and while originally designed in the context of link analysis on the web, both algorithms can be readily applied to citation patterns in academic papers and other citation graphs.

There are several aspects to the evaluation of a link analysis algorithm such as HITS or PageRank. One aspect relates to the specific notion of "authoritativeness" embodied by an algorithm. Thus specific users may have an understanding of what constitutes an authoritative web page or document in a given domain, and the output of HITS or PageRank can be evaluated by such users. While useful, such analyses often have a rather subjective flavor. A more objective criterion—the focus of the current paper—concerns the *stability* of a link analysis algorithm. Does an algorithm return similar results upon a small perturbation of the link structure or the document collection? We view stability as a desirable feature of a link analysis algorithm, above and beyond the particular notion of authoritativeness that the algorithm embodies. If an article is truly authoritative or influential, then surely the addition of a few links or a few citations should not make us change our minds about these sites or articles having been very influential. Moreover, even in the context of a fixed link structure, a dynamic, unreliable infrastructure such as the web may give us different views of the structure on different occasions. Ideally, a link analysis algorithm should be insensitive to such perturbations.

In this paper, we use techniques from matrix perturbation theory and coupled Markov chain theory to characterize the stability of the ranks assigned by HITS and PageRank. Some ways of improving the stability of HITS are also briefly metioned; these algorithmic changes are studied in more detail in [Ng *et al.*, 2001].

2 An Example

Let us begin with an empirical example. The *Cora* database [McCallum *et al.*, 2000] is a collection containing the citation information from several thousand papers in AI.

We ran the HITS and PageRank algorithms on the subset of the *Cora* database consisting of all its Machine Learning papers. To evaluate the stability of the two algorithms, we also constructed a set of five perturbed databases in which 30% of the papers from the base set were randomly deleted. ("Since *Cora* obtained its database via a web crawl, what if, by chance or mishap, it had instead retrieved only 70% of these papers?") If a paper is truly authoritative, we might hope that it would be possible to identify it as such with only a subset of the base set.

The results from HITS are shown in the following table. In this table, the first column reports the rank from HITS on the full set of Machine Learning papers, whereas the five rightmost columns report the ranks in runs on the perturbed databases. We see substantial variation across the different runs:

1	"Genetic algorithms in search, optimization...", Goldberg	1	3	1	1	1
2	"Adaptation in natural and artificial systems", Holland	2	5	3	3	2
3	"Genetic programming: On the programming of...", Koza	3	12	6	6	3
4	"Analysis of the behavior of a class of genetic...", De Jong	4	52	20	23	4
5	"Uniform crossover in genetic algorithms", Syswerda	5	171	119	99	5
6	"Artificial intelligence through simulated...", Fogel	6	135	56	40	8
7	"A survey of evolution strategies", Back+al	10	179	159	100	7
8	"Optimization of control parameters for genetic...", Grefenstette	8	316	141	170	6
9	"The GENITOR algorithm and selection pressure", Whitley	9	257	107	72	9
10	"Genetic algorithms + Data Structures = ...", Michalewicz	13	170	80	69	18
11	"Genetic programming II: Automatic discovey...", Koza	7	-	-	-	10
2060	"Learning internal representations by error...", Rumelhart+al	-	1	2	2	-

2061	"Learning to predict by the method of temporal...", Sutton	-	9	4	5	-
2063	"Some studies in machine learning using checkers", Samuel	-	-	10	10	-
2065	"Neuronlike elements that can solve difficult...", Barto+Sutton	-	-	8	-	-
2066	"Practical issues in TD learning", Tesauro	-	-	9	9	-
2071	"Pattern classification and scene analysis", Duda+Hart	-	4	7	7	-
2075	"Classification and regression trees", Breiman+al	-	2	5	4	-
2117	"UCI repository of machine learning databases", Murphy+Aha	-	7	-	8	-
2174	"Irrelevant features and the subset selection...", John+al	-	8	-	-	-
2184	"The CN2 induction algorithm", Clark+Niblett	-	6	-	-	-
2222	"Probabilistic reasoning in intelligent systems", Pearl	-	10	-	-	-

Although it might be thought that this variability is intrinsic to the problem, this is not the case, as shown by the results from the PageRank algorithm, which were much more stable:

1	"Genetic Algorithms in Search, Optimization and...", Goldberg	1	1	1	1	1
2	"Learning internal representations by error...", Rumelhart+al	2	2	2	2	2
3	"Adaptation in Natural and Artificial Systems", Holland	3	5	6	4	5
4	"Classification and Regression Trees", Breiman+al	4	3	5	5	4
5	"Probabilistic Reasoning in Intelligent Systems", Pearl	5	6	3	6	3
6	"Genetic Programming: On the Programming of ...", Koza	6	4	4	3	6
7	"Learning to Predict by the Methods of Temporal ...", Sutton	7	7	7	7	7
8	"Pattern classification and scene analysis", Duda+Hart	8	8	8	8	9
9	"Maximum likelihood from incomplete data via...", Dempster+al	10	9	9	11	8
10	"UCI repository of machine learning databases", Murphy+Aha	9	11	10	9	10
11	"Parallel Distributed Processing", Rumelhart+McClelland	-	-	-	10	-
12	"Introduction to the Theory of Neural Computation", Hertz+al	-	10	-	-	-

These results are discussed in more detail in Section 6. It should be stated at the outset, however, that our conclusion is not that HITS is unstable while PageRank is not. The issue is more subtle than that, involving considerations such as the relationships between multiple eigenvectors and invariant subspaces. We do wish to suggest, however, that stability is indeed an issue that needs attention. We now turn to a brief description of HITS and PageRank, followed by our analysis.

3 Overview of HITS and PageRank

Given a collection of web pages or academic papers linking to/citing each other, the HITS and PageRank algorithms each (implicitly) construct a matrix capturing the citation patterns, and determines authorities by computing the principal eigenvector of the matrix.[1]

3.1 HITS algorithm

The HITS algorithm [Kleinberg, 1998] posits that an article has high "authority" weight if it is linked to by many pages with high "hub" weight, and that a page has high hub weight if it links to many authoritative pages. More precisely, given a set of n web pages (say, retrieved in response to a search query), the HITS algorithm first forms the n-by-n adjacency matrix A, whose (i,j)-element is 1 if page i links to page j, and 0 otherwise.[2] It then iterates the following equations:

$$a_i^{(t+1)} = \sum_{j:j\to i} h_j^{(t)};\ h_i^{(t+1)} = \sum_{j:i\to j} a_j^{(t+1)}$$

(where "$i \to j$" means page i links to page j) to obtain the fixed-points $a^* = \lim_{t\to\infty} a^{(t)}$ and $h^* = \lim_{t\to\infty} h^{(t)}$ (with the vectors renormalized to unit length). The above equations can also be written:

$$\begin{aligned} a^{(t+1)} &= A^T h^{(t)} = (A^T A) a^{(t)} \\ h^{(t+1)} &= A a^{(t+1)} = (A A^T) h^{(t)}. \end{aligned}$$

When the iterations are initialized with the vector of ones $[1,\ldots,1]^T$, this is the power method of obtaining the principal eigenvector of a matrix [Golub and Van Loan, 1996], and so (under mild conditions) a^* and h^* are the principal eigenvectors of $A^T A$ and AA^T respectively. The "authoritativeness" of page i is then taken to be a_i^*, and likewise for hubs and h^*.

3.2 PageRank algorithm

Given a set of n web pages and the adjacency matrix A (defined previously), PageRank [Brin and Page, 1998] first constructs a probability transition matrix M by renormalizing each row of A to sum to 1. One then imagines a random web surfer who at each time step is at some web page, and decides which page to visit on the next step as follows: with probability $1-\epsilon$, she randomly picks one of the hyperlinks on the current page, and jumps to the page it links to; with probability ϵ, she "resets" by jumping to a web page picked uniformly and at random from the collection.[3] Here, ϵ is a parameter, typically set to 0.1-0.2. This process defines a Markov chain on the web pages, with transition matrix $\epsilon U + (1-\epsilon)M$, where U is the transition matrix of uniform transition probabilities ($U_{ij} = 1/n$ for all i,j). The vector of PageRank scores p is then defined to be the stationary distribution of this Markov chain. Equivalently, p is the principal eigenvector of the transition matrix $(\epsilon U + (1-\epsilon)M)^T$ (see, e.g. Golub and Van Loan, 1996), since by definition the stationary distribution satisfies

$$(\epsilon U + (1-\epsilon)M)^T p = p. \qquad (1)$$

The asymptotic chance of visiting page i, that is, p_i, is then taken to be the "quality" or authoritativeness of page i.

4 Analysis of Algorithms

We begin with a simple example showing how a small addition to a collection of web pages can result in a large change to the eigenvectors returned. Suppose we have a collection of web pages that contains 100 web pages linking to `http://www.algore.com`, and another 103 web pages

[1] It is worth noting that HITS is typically described as running on a small collection of articles (say retrieved in response to a query), while PageRank is described in terms of the entire web. Either algorithm can be run in either setting, however, and this distinction plays no role in our analysis.

[2] Kleinberg [1998] discusses several other heuristics regarding issues such as intra-domain references, which are ignored in this section for simplicity (but are used in our experiments). See also Bharat and Henzinger [1998] for other improvements to HITS. It should be noted that none of these fundamentally change the spirit of the eigenvector calculations underlying HITS.

[3] There are various ways to treat the case of pages with no outlinks (leaf nodes). In this paper we utilize a particularly simple approach—upon reaching such a page, the web surfer picks the next page uniformly at random. This means that if a row of A has all zero entries, then the corresponding row of M is constructed to have all entries equal to $1/n$. The PageRank algorithm described in [Page *et al.*, 1998] utilizes a different reset distribution upon arriving at a leaf node. It is possible to show, however, that every instantiation of our variant of the algorithm is equivalent to an instantiation of the original algorithm on the same graph with a different value of the reset probability.

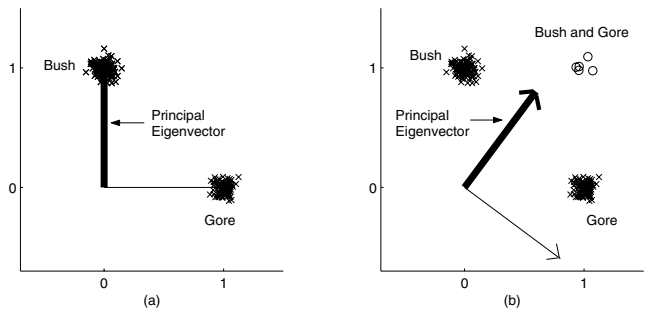


Figure 1: Jittered scatterplot of hyperlink graph.

linking to `http://www.georgewbush.com`. The adjacency matrix A has all zeros except for the two columns corresponding to these two web pages, therefore the principal eigenvector a^* will have non-zero values only for `algore.com` and `georgewbush.com`. Figure 1(a) presents a jittered scatterplot of links to these two web pages, along with the first two eigenvectors. (Only the non-zero portions of the eigenvectors are shown.) Now, suppose five new web pages trickle into our collection, which happen to link to both `algore.com` and `georgewbush.com`. Figure 1(b) shows the new plot, and we see that the eigenvectors have changed dramatically, with the principal eigenvector now near the $45°$ line. Thus, a relatively *small* perturbation to our collection has caused a *large* change to the eigenvectors.[4] If this phenomenon is pervasive, then it needs to be addressed by any algorithm that uses eigenvectors to determine authority. In the next two sections, we give characterizations of whether and when algorithms can be expected to suffer from these problems.

4.1 Analysis of HITS

HITS uses the principal eigenvector of $S = A^T A$ to determine authorities. In this section, we show that the stability of this eigenvector under small perturbations is determined by the *eigengap* of S, which is defined to be the difference between the largest and the second largest eigenvalues.

Here is an example that may shed light on the importance of the eigengap. Figure 2 plots the contours associated with two matrices S_1 and S_2 before (with solid lines) and after (with dashed lines) the same additive perturbation have been made to them.[5] The eigenvalues of the matrices are indicated by the directions of the principal axes of the ellipses. The matrix S_1 shown in Figure 2a has eigengap $\delta_1 \approx 0$, and a small perturbation to S_1 (and hence the ellipse) results in eigenvectors $45°$ away from the original eigenvectors; the matrix S_2 shown in Figure 2b has eigengap $\delta_2 = 2$, and the perturbed eigenvectors are nearly the same as the original eigenvectors. So, we see how, in this example, the size of the eigengap directly affects the stability of the eigenvectors. (Readers familiar with plots of multivariate Gaussians can also think of these as the contours of a Gaussian with small perturbations imposed on the (inverse) covariance matrix.)

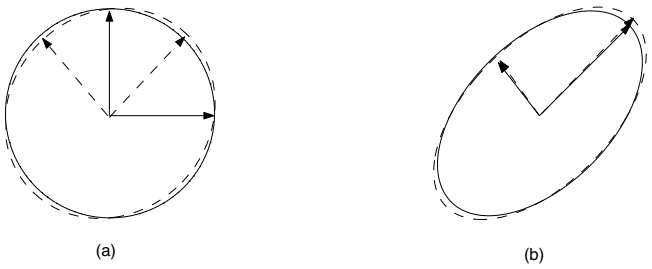


Figure 2: Contours of two matrices with different eigengaps.

In the sequel, we use a tilde to denote perturbed quantities. (For instance, $\tilde{S}$ denotes a perturbed version of S.) We now give our first, positive result, that so long as the eigengap δ is large, then HITS is insensitive to small perturbations.[6]

Theorem 1. *Let $S = A^T A$ be given. Let a^* be the principal eigenvector and δ the eigengap of S. Assume the maximum out-degree of every web page is bounded by d. For any $\varepsilon > 0$, suppose we perturb the web/citation graph by adding or deleting at most k links from one page, where $k < (\sqrt{d+\alpha} - \sqrt{d})^2$, where $\alpha = \varepsilon\delta/(4+\sqrt{2}\varepsilon)$. Then the perturbed principal eigenvector $\tilde{a}^*$ of the perturbed matrix $\tilde{S}$ satisfies:*

$$||a^* - \tilde{a}^*||_2 \leq \varepsilon \tag{2}$$

So, if the eigengap is big, HITS will be insensitive to small perturbations. This result is proved by showing i) the *direction* of the principal eigenvector does not change too much, and ii) the *magnitudes* of the relevant eigenvalues do not change too much, so the second eigenvector does not "overtake" the first and become the new principal eigenvector.

Proof. Let $||\cdot||_F$ denote the Frobenius norm.[7] We apply Theorem V.2.8 from matrix perturbation theory [Stewart and Sun, 1990]: Suppose $S \in \mathbb{R}^{n\times n}$ is a symmetric matrix with principal eigenvalue λ^* and eigenvector a^*, and eigengap $\delta > 0$. Let E be a symmetric perturbation to S. Then the following inequalities hold for the old principal eigenpair (λ^*, a^*) and *some* new eigenpair $(\tilde{\lambda}, \tilde{a})$.

$$||a^* - \tilde{a}||_2 \leq \frac{4||E||_F}{\delta - \sqrt{2}||E||_F} \tag{3}$$

$$|\lambda^* - \tilde{\lambda}| \leq \sqrt{2}||E||_F \tag{4}$$

(assuming that the denominator in (3) is positive). Let the complementary eigenspace to (λ^*, a^*) be represented by (L_2, X_2), i.e. X_2 is orthonormal, and its columns contain all the eigenvectors of S except a^*; L_2 is diagonal and contains the corresponding eigenvalues, all of which are at least δ less

[4] There is nothing special about the number 5 here; a smaller number also results in relatively large swings of the eigenvectors. Replacing 5 with 1, 2, 3, and 4 causes the principal eigenvector to lie at 73, 63, 58 and 55 degrees, respectively.

[5] More precisely, these are contours of the quadratic form $x^T S_i x$.

[6] Our analyses also apply directly to hub-weight calculations, simply by reversing link directions and interchanging A and A^T.

[7] The Frobenius norm is defined by $||X||_F = (\sum_i \sum_j (X_{ij})^2)^{1/2}$.

than λ^*; and $SX_2 = X_2L_2$. A bound similar to Equation (4) holds for L_2:

$$||L_2 - \tilde{L}_2||_F \leq \sqrt{2}||E||_F \tag{5}$$

Let $\tilde{\lambda}_2$ be the largest eigenvalue of $\tilde{L}_2$. Using Corollary IV.3.6 from Stewart and Sun [1990], one can show that Equation (5) implies

$$\tilde{\lambda}_2 \leq \lambda_2 + \sqrt{2}||E||_F \tag{6}$$

If in turn $\sqrt{2}||E||_F < \delta/2$, then Equations (4) and (6) together will ensure that $\tilde{\lambda} > \tilde{\lambda}_2$, i.e. $(\tilde{\lambda}, \tilde{a})$ is the principal eigenpair of $\tilde{S}$.

Since we are adding or deleting links from only one page, let F denote the perturbation to one row of A, so that $\tilde{S} = (A+F)^T(A+F)$. It is straightforward to show $||F^TF||_F \leq k$ and $||A^TF||_F = ||F^TA||_F \leq \sqrt{dk}$. We can thus bound the norm of the perturbation to S:

$$||E||_F = ||\tilde{S} - S||_F \leq k + 2\sqrt{dk} \tag{7}$$

Using Equations (3) and (7) to determine when we may guarantee Equation (2) to hold, we arrive at the bound $k < (\sqrt{d+\alpha} - \sqrt{d})^2$, where $\alpha = \varepsilon\delta/(4+\sqrt{2}\varepsilon)$. One can easily verify that the same bound on k also ensures $\sqrt{2}||E||_F < \delta/2$ (which also guarantees that the denominator in (3) is positive), hence $\tilde{a}^* = \tilde{a}$ as previously stated. □

Next we give the converse of this result, that if the eigengap is small, then eigenvectors can be sensitive to perturbations.

Theorem 2. *Suppose S is a symmetric matrix with eigengap δ. Then there exists a $O(\delta)$ perturbation[8] to S that causes a large ($\Omega(1)$) change in the principal eigenvector.*

Proof. Since $S = S^T$, it can be diagonalized:

$$S = U \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \Sigma \end{pmatrix} U^T$$

where U is orthogonal, and whose columns are the S's eigenvectors. Let u_i denote the i-th column of U. We pick $\tilde{S} = S + 2\delta u_2 u_2^T$. Since $||u_2||_2 = 1$, the norm of the perturbation is only $||2\delta u_2 u_2^T||_F = 2\delta$. Moreover,

$$\tilde{S} = U \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 + 2\delta & 0 \\ 0 & 0 & \Sigma \end{pmatrix} U^T$$

As $\tilde{\lambda}_2 = \lambda_2 + 2\delta > \lambda_1$, $(\tilde{\lambda}_2, u_2)$ is the new principal eigenpair. But u_2 is orthogonal to u_1, so $||u_2 - u_1||_2 = \Omega(1)$. □

To ground these results and illustrate why Theorem 1 requires a bound d on out-degrees, we give another example of where a small perturbation—adding a single link—can have a large effect. In this example we use the fact that if a graph has multiple connected components, then the principal eigenvalue will have non-zero entries in nodes only from the "largest" connected component (more formally the component with the largest eigenvalue).[9]

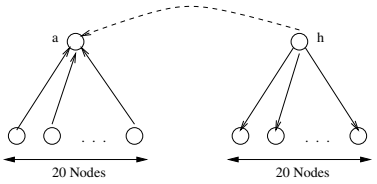


Figure 3: Picture of a web community.

Consider the web/citation-graph shown in Figure 3, which we imagine to be a small subset of a much larger graph. Solid arrows denote the original set of hyperlinks; the dashed arrow represents the link we will add. The original principal eigenvalue for each of the two connected components shown is $\lambda = 20$; with the addition of a single link, it is easy to verify that this jumps to $\tilde{\lambda} = 25$. Suppose that the community shown is part of a larger web/citation graph with multiple subcommunities, and that originally the biggest subcommunity had eigenvalue $20 < \lambda_1 < 25$. By adding one link, the graph shown in Figure 3 becomes the biggest subcommunity, and the principal eigenvector now has positive values only for nodes shown in this figure, and zeros elsewhere.

4.2 Analysis of PageRank

We now analyze the sensitivity of PageRank's authority scores p to perturbations of the web/citation-graph.

Theorem 3. *Let M be given, and let p be the principal right eigenvector of $(\epsilon U + (1-\epsilon)M)^T$. Let articles/pages $i_1, i_2, \ldots, i_k$ be changed in any way, and $\tilde{M}$ be the corresponding (new) transition matrix. Then the new PageRank scores $\tilde{p}$ satisfies:*

$$||\tilde{p} - p||_1 \leq \frac{2\sum_{j=1}^{k} p_{i_j}}{\epsilon} \tag{8}$$

Thus, assuming ϵ is not too close to 0, this shows that so long as the perturbed/modified web pages did not have high overall PageRank scores (as measured with respect to the *unperturbed* PageRank scores p), then the perturbed PageRank scores $\tilde{p}$ will not be far from the original.

Proof. We construct a coupled Markov chain $\{(X_t, Y_t) : t \geq 0\}$ over pairs of web pages/documents as follows. $X_0 = Y_0$ is drawn according to the probability vector p, that is, from the stationary distribution of the PageRank "random surfer" model. The state transitions work as follows: On step t, we decide with probability ϵ to "reset" both chains, in which case we set X_t and Y_t to the *same* page chosen uniformly at random from the collection. If no "reset" occurs, and if $X_{t-1} = Y_{t-1}$ and X_{t-1} is one of the unperturbed pages, then $X_t = Y_t$ is chosen to be a random page linked to by the page X_{t-1}. In all other cases, X_t is chosen to be a random page linked to by page X_{t-1}, and independently of it, Y_t is chosen to be a random page linked to by page Y_{t-1}.

[8] More formally, there exists a perturbed version of S, denoted $\tilde{S}$, so that $||S - \tilde{S}||_F = O(\delta)$.

[9] See, e.g. Chung [1994]. A connected component of a graph is a subset whose elements are connected via length ≥ 1 paths to each other, but not to the rest of the graph. The eigenvalue of a connected component C is the largest eigenvalue of $A_C^T A_C$ (cf. A^TA used by HITS), where A_C, a submatrix of A, is the adjacency matrix of C.

Thus, we now have two "coupled" Markov chains X_t and Y_t, the former using the transition probabilities $(\epsilon U + (1-\epsilon)M)^T$, and latter $(\epsilon U + (1-\epsilon)\tilde{M})^T$, but so that their transitions are "correlated." For instance, the "resets" to both chains always occur in lock-step. But since each chain is following its own state transition distribution, the asymptotic distributions of X_t and Y_t must respectively be p and $\tilde{p}$. Now, let $d_t = P(X_t \neq Y_t)$. Note $d_0 = 0$, since $X_0 = Y_0$ always. Letting $\mathcal{P}$ denote the set of perturbed pages, we have:

$$\begin{aligned}
d_{t+1} &= P(X_{t+1} \neq Y_{t+1}) \\
&= P(X_{t+1} \neq Y_{t+1} | \text{reset at } t+1) P(\text{reset}) \\
&\quad + P(X_{t+1} \neq Y_{t+1} | \text{no reset at } t+1) P(\text{no reset}) \\
&= 0 \cdot \epsilon + (1-\epsilon) P(X_{t+1} \neq Y_{t+1} | \text{no reset at } t+1) \\
&= (1-\epsilon)[P(X_{t+1} \neq Y_{t+1}, X_t \neq Y_t | \text{no reset at } t+1) \\
&\quad + P(X_{t+1} \neq Y_{t+1}, X_t = Y_t | \text{no reset at } t+1)] \\
&\leq (1-\epsilon)[P(X_t \neq Y_t | \text{no reset at } t+1) \\
&\quad + P(X_{t+1} \neq Y_{t+1}, X_t = Y_t, X_t \in \mathcal{P} | \text{no reset at } t+1)] \\
&\leq (1-\epsilon)(P(X_t \neq Y_t) + P(X_t \in \mathcal{P} | \text{no reset at } t+1)) \\
&\leq (1-\epsilon)(d_t + \textstyle\sum_{i \in \mathcal{P}} p_i)
\end{aligned}$$

where to derive the first inequality, we used the fact that by construction, the event "$X_{t+1} \neq Y_{t+1}, X_t = Y_t$" is possible only if X_t is one of the perturbed pages. Using the fact that $d_0 = 0$ and by iterating this bound on d_{t+1} in terms of d_t, we obtain an asymptotic upper-bound: $d_\infty \leq (\sum_{i \in \mathcal{P}} p_i)/\epsilon$. Thus, if (X_∞, Y_∞) is drawn from the stationary distribution of the correlated chains—so the marginal distributions of X_∞ and Y_∞ are respectively given by p and $\tilde{p}$—then $P(X_\infty \neq Y_\infty) = d_\infty \leq (\sum_{i \in \mathcal{P}} p_i)/\epsilon$. But if two random variables have only a small d_∞ chance of taking different values, then their distributions must be similar. More precisely, by the Coupling Lemma (e.g., see Aldous, 1983) the *variational distance* $(1/2)\sum_i |p_i - \tilde{p}_i|$ between the distributions must also be bounded by the same quantity d_∞. This shows $||p - \tilde{p}||_1 \leq 2d_\infty$, which concludes the proof. □

5 LSI and HITS

In this section we present an interesting connection between HITS and Latent Semantic Indexing [Deerwester *et al.*, 1990] (LSI) that provides additional insight into our stability results (see also Cohn and Chang, 2000). In LSI a collection of documents is represented as a matrix A, where A_{ij} is 1 if document j contains the i-th word of the vocabulary, and 0 otherwise. LSI computes the left and right singular vectors of A (equivalently, the eigenvectors of AA^T and A^TA). For example, the principal left singular vector, which we denote x, has dimension equal to the vocabulary size, and x_j measures the "strength" of word j's membership along the x-dimension. The informal hope is that synonyms will be grouped into the same singular vectors, so that when a document (represented by a column of A) is projected onto the subspace spanned by the singular vectors, it will automatically be "expanded" to include synonyms of words in the document, leading to improved information retrieval.

Now consider constructing the following citation graph from a set of documents. Let there be a node for each document and for each word. The node of a word links to the document nodes it appears in. Let $\hat{A}$ be the adjacency matrix of this graph. If we apply HITS to this graph, we find only the word-nodes have non-zero hub weights (since none of the document-nodes link to anything) and only the document-nodes have non-zero authority weights. Moreover, the vector of HITS hub weights of the word-nodes is exactly x, the first left singular vector found by LSI.

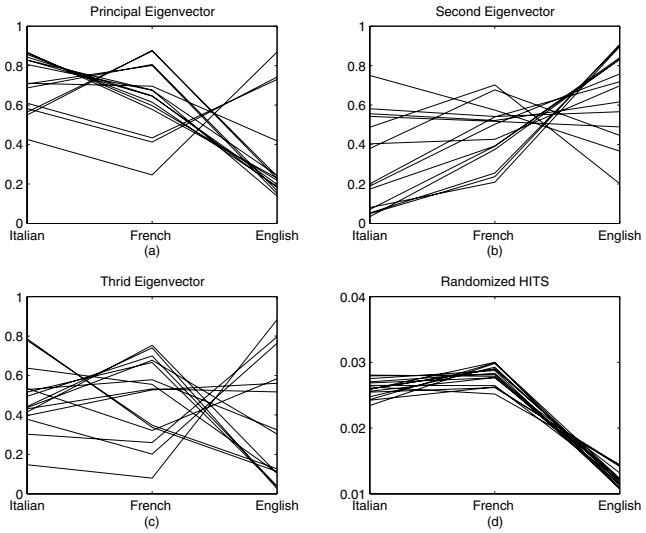


Figure 4: Results on random corpora.

This connection allows us to transfer insight from experiments on LSI to our understanding of HITS. In this vein, we conducted an experiment in which random corpora were generated by sampling from a set of English, French, and Italian documents.[10] Given that these random corpora are combinations of three distinct languages, the solution to Information Retrieval problems such as clustering or synonym-identification are exceedingly simple. The issue that we are interested in, however, is stability. To study stability, we generated 15 such collections and examined the direction of the principal eigenvectors found by HITS.

The principal eigenvector lies in the high dimensional joint-vocabulary space of the three languages. To display our results, we therefore defined English, French, and Italian "directions," and measured the degree to which the eigenvector lies in each these directions.[11] Fifteen independent repetitions of this process were carried out, and the results plotted in Figure 4a. As we see, despite the presence of clear clusters in the corpora, the eigenvectors are highly variable. Moreover, this variability persists in the second and third eigenvectors (Figures 4b,c).

[10] The corpora were generated by taking paragraphs from novels in the three languages. Typical "documents" had 25–150 words, and the vocabulary consisted of the most common 1500 words per language. The collection was also manually "balanced" to equally represent each language.

[11] This was done by picking a vector x_e of unit-norm and whose i-th element is proportional to the frequency of word i in the English collection—thus, x_e should be thought of as the "canonical" English direction—and taking the amount that h^* lies in the English direction to be the absolute magnitude of the dot-product between x_e and h^*, and similarly for French and Italian.

Note that the variability is not an inherent feature of the problem. In Figure 4d, we display a run of a different algorithm (a variant of the HITS algorithm that we briefly describe in Section 7, and is studied in more detail in [Ng *et al.*, 2001]). Here the results are significantly less variable.

6 Further Experiments

In this section we report further results of perturbation experiments on the *Cora* database. We also describe an experiment using web pages.

Recall our methodology in the experiments with the *Cora* database: We choose a subset of papers from the database and generate a set of perturbations to this subset by randomly deleting 30% of the papers. Our first experiment used all of the AI papers in Cora as the base set. Our results largely replicated those of Cohn and Chang [2000]—HITS returned several Genetics Algorithms (GA) papers as the top-ranked ones. With the database perturbed as described, however, these results were very variable, and HITS often returned seminal papers from broader AI areas as its top-ranked documents. Repeating the experiment excluding all the GA papers, HITS did slightly better; the results on five independent trials are shown below:

1	"Classification and Regression Trees", Brieman+al	1	1	1	1	1
2	"Pattern classification and scene analysis", Duda+Hart	2	2	3	2	2
3	"UCI repository of machine learning databases", Murphy+Aha	4	3	7	3	3
4	"Learning internal representations by error...", Rumelhart+al	3	13	2	28	20
5	"Irrelevant Features and the Subset Selection Problem", John+al	7	4	12	4	4
6	"Very simple classification rules perform well on...", Holte	8	5	15	5	5
7	"C4.5: Programs for Machine Learning", Quinlan	11	10	14	10	6
8	"Probabilistic Reasoning in Intelligent Systems", Pearl	6	459	4	462	461
9	"The CN2 induction algorithm", Clark+Niblett	9	54	11	78	105
10	"Learning Boolean Concepts in the ...", Almuallim+Dietterich	14	11	34	9	13
11	"The MONK's problems: A performance comparison...", Thrun	-	9	-	6	7
12	"Inferring decision trees using the MDL Principle", Quinlan	-	8	-	7	8
13	"Multi-interval discretization of continuous...", Fayyad+Irani	-	-	-	-	10
14	"Learning Relations by Pathfinding", Richards+Moon	-	6	-	-	-
15	"A conservation law for generalization performance", Schaffer	-	7	-	8	-
20	"The Feature Selection Problem: Traditional..." Kira+Randall	-	-	-	-	9
21	"Maximum likelihood from incomplete data via..." Dempster+al	10	-	5	-	-
23	"Learning to Predict by the Method of Temporal...", Sutton	5	-	6	-	-
36	"Introduction to the Theory of Neural Computation", Hertz+al	-	-	8	-	-
49	"Explanation-based generalization: a unifying view", Mitchell	-	-	10	-	-
282	"A robust layered control system for a mobile robot", Brooks	-	-	9	-	-

We see that, apart from the top 2-3 ranked papers, the remaining results are still rather unstable. For example, Pearl's book was originally ranked 8th; on the second trial, it dropped to rank 459. Similarly, Brooks' paper was rank 282, and jumped up to rank 9 on trial 3. However, this variability is not intrinsic to the problem, as shown by our PageRank results (all PageRank results in this section were generated with $\epsilon = 0.2$):

1	"Classification and Regression Trees", Breiman+al	1	1	1	1	2
2	"Probabilistic Reasoning in Intelligent Systems", Pearl	3	2	2	2	1
3	"Learning internal representations by error...", Rumelhart+al	2	3	3	3	3
4	"Pattern classification and scene analysis", Duda+Hart	4	4	4	4	4
5	"A robust layered control system for a mobile robot", Brooks	5	6	7	5	5
6	"Maximum likelihood from incomplete data via...' Dempster+al	6	7	6	6	6
7	"Learning to Predict by the Method of Temporal...", Sutton	7	5	5	7	7
8	"UCI repository of machine learning databases", Murphy+Aha	8	9	9	9	11
9	"Numerical Recipes in C", Press+al	10	12	8	11	8
10	"Parallel Distributed Processing", Rumelhart+al	9	14	13	10	9
12	"An implementation of a theory of activity", Agre+Chapmanre	-	8	10	8	-
13	"Introduction to the Theory of Neural Computation", Hertz+al	-	10	-	-	-
22	"A Representation and Library for Objectives in...", Valente+al	-	-	-	-	10

The largest change in a document's rank was a drop from 10 to 12—these results are much more stable than for HITS. Closer examination of the HITS authority weights reviews that its jumps in rankings are indeed due to large changes in authority weights, whereas the PageRank scores tended to remain fairly stable.[12]

We also carried out experiments on web pages. Given a query, Kleinberg [1998] describes a method for obtaining a collection of web pages on which to run HITS. We use exactly the method described there, and perturbed it in a natural way.[13] For the sake of brevity, we only give the results of two experiments here. On the query "mp3 players", HITS' results were as follows (long URLs are truncated):

1	http://www.freecode.com/	82	1	1	1	82
2	http://www.htmlworks.com/	85	2	2	2	83
3	http://www.internettrafficreport.com/	86	3	4	3	85
4	http://slashdot.org/	88	4	5	5	86
5	http://windows.davecentral.com/	87	5	3	4	84
6	http://www.gifworks.com/	84	6	6	6	87
7	http://www.thinkgeek.com/	91	7	7	7	88
8	http://www.animfactory.com/	89	9	8	8	89
9	http://freshmeat.net/	90	8	9	9	90
10	http://subscribe.andover.net/membership.htm	92	10	10	10	91
1385	http://ourstory.about.com/index.htm	1	-	-	-	1
1386	http://home.about.com/index.htm	2	-	-	-	2
1387	http://home.about.com/musicperform/index.htm	3	-	-	-	3
1388	http://home.about.com/teens/index.htm	4	-	-	-	4
1389	http://home.about.com/sports/index.htm	5	-	-	-	5
1390	http://home.about.com/autos/index.htm	6	-	-	-	6
1391	http://home.about.com/style/index.htm	7	-	-	-	7
1392	http://home.about.com/careers/index.htm	8	-	-	-	8
1393	http://home.about.com/citiestowns/index.htm	9	-	-	-	9
1394	http://home.about.com/travel/index.htm	10	-	-	-	10

In contrast, PageRank returned:

1	http://www.team-mp3.com/	*	1	1	1	1
2	http://click.linksynergy.com/fs-bin/click	1	3	2	4	9
3	http://www.elizandra.com/	2	2	3	2	2
4	http://stores.yahoo.com/help.html	4	14	5	10	11
5	http://shopping.yahoo.com/	3	10	4	12	13
6	http://www.netins.net/showcase/phdss/	*	8	6	3	3
7	http://www.thecounter.com/	13	6	9	8	7
8	http://ourstory.about.com/index.htm	5	4	7	5	4
9	http://a-zlist.about.com/index.htm	6	5	10	6	6
10	http://www.netins.net/showcase/phdss/getm	*	9	8	7	5
11	http://software.mp3.com/software/	7	7	-	-	8
12	http://www.winamp.com/	8	-	-	-	-
13	http://www.nullsoft.com/	10	-	-	-	-
14	http://www.consumerspot.com/redirect/1cac	9	-	-	9	10

While PageRank's rankings undergo small changes, HITS' rankings display a mass "flipping" behavior. Similar perturbation patterns to this (and the example below) for PageRank and HITS are observed in fourteen out of nineteen queries. Furthermore, HITS' results displayed such mass "flips" in roughly 20% of the trials, which is in accordance with the 20% removal rate.

Here is another typical web result, this time on the query "italian recipes." Note that "*" means that the page was removed by that trial's perturbation, and therefore has no rank. HITS' results were:

[12] Examination of the second and higher eigenvectors in HITS shows that they, too, can vary substantially from trial to trial.

[13] Kleinberg [1998] first uses a web search engine (www.altavista.com in our case) to retrieve 200 documents to form a "root set," which is then expanded (and further processed) to define the web-graph on which HITS operates. Our perturbations were arrived at by randomly deleting 20% of the root set (i.e. imagining that the web search engine had only returned 80% of the pages it actually did), and then following Kleinberg's procedure.

1	http://ourstory.about.com/index.htm	*	1	1	1	1
2	http://home.about.com/culture/index.htm	*	2	2	2	17
3	http://home.about.com/index.htm	*	3	3	3	25
4	http://home.about.com/food/index.htm	*	4	4	4	2
5	http://home.about.com/science/index.htm	*	5	5	5	3
6	http://home.about.com/shopping/index.htm	*	6	6	6	4
7	http://home.about.com/smallbusiness/index	*	7	7	7	5
8	http://home.about.com/sports/index.htm	*	8	8	8	6
9	http://home.about.com/arts/index.htm	*	9	9	9	7
10	http://home.about.com/style/index.htm	*	10	10	10	8
11	http://home.about.com/autos/index.htm	-	-	-	-	9
12	http://home.about.com/teens/index.htm	-	-	-	-	10
479	http://bestbrandrecipe.com/default.asp	1	-	-	-	-
480	http://myrecipe.com/help/shopping.asp	2	-	-	-	-
481	http://vegetarianrecipe.com/default.asp	3	-	-	-	-
482	http://holidayrecipe.com/default.asp	5	-	-	-	-
483	http://beefrecipe.com/default.asp	4	-	-	-	-
484	http://beveragerecipe.com/default.asp	7	-	-	-	-
485	http://appetizerrecipe.com/default.asp	6	-	-	-	-
486	http://pierecipe.com/default.asp	8	-	-	-	-
487	http://seafoodrecipe.com/default.asp	9	-	-	-	-
488	http://barbequerecipe.com/default.asp	10	-	-	-	-

PageRank, on the other hand, returned:

1	http://ourstory.about.com/index.htm	*	1	1	1	1
2	http://a-zlist.about.com/index.htm	*	2	2	2	2
3	http://www.apple.com/	1	3	3	3	3
4	http://www.tznet.com/isenberg/	2	4	4	13	9
5	http://frontier.userland.com/	3	5	5	4	7
6	http://www.mikrostore.com/	4	6	6	5	*
7	http://www.amazinggiftsonline.com/	5	7	7	6	*
8	http://www.peck.it/peckshop/home.asp?prov	*	8	8	7	4
9	http://geocities.yahoo.com/addons/interac	6	9	9	8	29
10	http://dvs.dolcevita.com/index.html	7	10	*	10	5
11	http://www.dossier.net/	-	-	10	9	6
12	http://www.dolcevita.com/	8	-	-	-	8
14	http://www.q-d.com/	9	-	-	-	10
15	http://www.silesky.com/	10	-	-	-	-

7 Discussion

It is well known in the numerical linear algebra community that a subspace spanned by several (e.g. the first k) eigenvectors may be stable under perturbation, while individual eigenvectors may not [Stewart and Sun, 1990]. Our results—both theoretical and empirical—reflect this general fact.

If the output of an algorithm is a subspace, then the stability considerations that we have discussed may not be a matter of primary concern. Such is the case, for example, for the LSI algorithm, where the goal is generally to project a data set onto a lower-dimensional subspace.

If we wish to interpret specific eigenvectors, however, then the stability issue becomes a matter of more serious concern. This is the situation for the basic HITS algorithm, where primary eigenvectors have been interpreted in terms of a set of "hubs" and "authorities." As we have seen, there are theoretical and empirical reasons for exercising considerable caution in making such interpretations.

Given that the principal eigenvector may not have a reliable interpretation, one can consider variations of the HITS approach that utilize multiple eigenvectors. Indeed, Kleinberg [1998] suggested examining multiple eigenvectors as a way of obtaining authorities within multiple communities. Again, however, it may be problematic to interpret individual eigenvectors, and in fact in our experiments we found significant variability in second and third eigenvectors. An alternative approach may be to automatically combine multiple eigenvectors in a way that explicitly identifies subspaces within the HITS framework. This is explored in [Ng *et al.*, 2001]

The fact that the PageRank algorithm appears to be relatively immune to stability concerns is a matter of considerable interest. It is our belief that the "reset-to-uniform-distribution" aspect of PageRank is a critical feature in this regard. Indeed, one can explore a variation of the HITS algorithm which incorporates such a feature. Suppose that we construct a Markov chain on the web in which, with probability $1 - \epsilon$, we randomly follow a hyperlink from the current page in the forward direction (on odd time steps), and we randomly follow a hyperlink in the backwards direction (on even time steps). With probability ϵ, we reset to a uniformly chosen page. The asymptotic web-page visitation distribution on odd steps is defined to be the authority weights, and on even steps the hub weights. As in Theorem 3, we can show this algorithm is insensitive to small perturbations (but unlike PageRank, we obtain hub as well as authority scores). The results of running this algorithm on the "three languages" problem are shown in Figure 4d, where we see that it is indeed significantly more stable than the basic HITS algorithm. This algorithm is also explored in more detail in [Ng *et al.*, 2001].

Acknowledgments

We thank Andrew McCallum for providing the Cora citation data used in our experiments. We also thank Andy Zimdars for helpful comments. This work was supported by ONR MURI N00014-00-1-0637 and NSF grant IIS-9988642.

References

[Aldous, 1983] David Aldous. Random walks on finite groups and rapidly mixing markov chains. In A. Dold and B. Eckmann, editors, *Séminaire de Probabilités XVII 1981/1982*, Lecture Notes in Mathematics, Vol. 986, pages 243–297. Springer-Verlag, 1983.

[Bharat and Henzinger, 1998] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proc. 21st Annual Intl. ACM SIGIR Conference*, pages 104–111. ACM, 1998.

[Brin and Page, 1998] S. Brin and L. Page. The anatomy of a large-scale hypertextual (Web) search engine. In *The Seventh International World Wide Web Conference*, 1998.

[Chung, 1994] Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1994.

[Cohn and Chang, 2000] D. Cohn and H. Chang. Probabilistically identifying authoritative documents. In *Proc. 17th International Conference on Machine Learning*, 2000.

[Deerwester *et al.*, 1990] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[Golub and Van Loan, 1996] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Univ. Press, 1996.

[Kleinberg, 1998] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[McCallum *et al.*, 2000] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the contruction of Internet portals with machine learning. *Information Retrieval Journal*, 3:127–163, 2000.

[Ng *et al.*, 2001] Andrew Y. Ng, Alice X. Zheng, and Michael I. Jordan. Stable algorithms for link analysis. In *Proc. 24th Annual Intl. ACM SIGIR Conference*. ACM, 2001.

[Osareh, 1996] Farideh Osareh. Bibliometrics, citation analysis and co-citation analysis: A review of literature I. *Libri*, 46:149–158, 1996.

[Page *et al.*, 1998] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Unpublished Manuscript, 1998.

[Stewart and Sun, 1990] G. W. Stewart and Ji-Guang Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

Active Learning for Class Probability Estimation and Ranking

Maytal Saar-Tsechansky and Foster Provost
Department of Information Systems
Leonard N. Stern School of Business, New York University
{mtsechan|fprovost}@stern.nyu.edu

Abstract

For many supervised learning tasks it is very costly to produce training data with class labels. *Active learning* acquires data incrementally, at each stage using the model learned so far to help identify especially useful additional data for labeling. Existing empirical active learning approaches have focused on learning classifiers. However, many applications require estimations of the probability of class membership, or scores that can be used to rank new cases. We present a new active learning method for class probability estimation (CPE) and ranking. BOOTSTRAP-LV selects new data for labeling based on the variance in probability estimates, as determined by learning multiple models from bootstrap samples of the existing labeled data. We show empirically that the method reduces the number of data items that must be labeled, across a wide variety of data sets. We also compare BOOTSTRAP-LV with UNCERTAINTY SAMPLING, an existing active learning method designed to maximize classification accuracy. The results show that BOOTSTRAP-LV dominates for CPE. Surprisingly it also often is preferable for accelerating simple accuracy maximization.

1 Introduction

Supervised classifier learning requires data with class labels. In many applications, procuring class labels can be costly. For example, to learn diagnostic models experts may need to analyze many historical cases. To learn document classifiers experts may need read many documents and assign them labels. To learn customer response models, consumers may have to be given costly incentives to reveal their preferences.

Active learning processes training data incrementally, using the model learned "so far" to select particularly helpful additional training examples for labeling. When successful, active learning methods reduce the number of instances that must be labeled to achieve a particular level of accuracy. Most existing methods and particularly empirical approaches for active learning address classification problems—they assume the task is to assign cases to one of a fixed number of classes.

Many applications require more than simple classification. Decision-making often requires estimates of the probability of class membership. Class probability estimates (CPEs) can be combined with decision-making costs/benefits to minimize expected cost (maximize expected benefit). For example, in target marketing the estimated probability that a customer will respond to an offer is combined with the estimated profit (produced with a different model) [Zadrozny and Elkan, 2001]. Other applications require ranking of cases, to add flexibility to user processing.[1] We agree with Turney [Turney, 2000] that machine learning systems should be able to take into account *various* cost/benefit information, including decision-making costs as well as labeling costs.

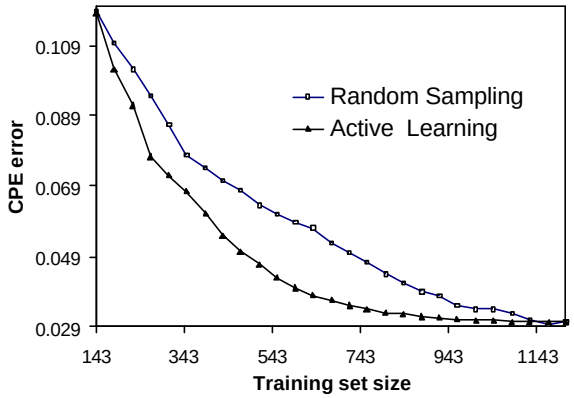


Figure 1: Learning curves for the Car data set

In this paper, we consider active learning to produce accurate CPEs and class-based rankings. Figure 1 shows the desired behavior of an active learner. The horizontal axis represents the number of training data, and the vertical axis represents the error rate of the probabilities produced by the model learned. Each *learning curve* shows how error rate decreases as more training data are used. The upper curve represents the decrease in error from randomly

[1] Classification accuracy has been criticized previously as a metric for machine learning research (Provost et al., 1998).

selecting training data; the lower curve represents active learning. The two curves form a "banana" shape: very early on, the curves are comparable because a model is not yet available for active learning. The active learning curve soon accelerates, because of the careful choice of training data. Given enough data, random selection catches up.

We introduce a new active learning technique, BOOTSTRAP-LV, which uses bootstrap samples of existing training data to examine the variance in the probability estimates for not-yet-labeled data. We show empirically across a wide range of data sets that BOOTSTRAP-LV decreases the number of labeled instances needed to achieve accurate probability estimates, or alternatively that it increases the accuracy of the probability estimates for a fixed number of training data. We also show that BOOTSTRAP-LV is surprisingly effective even for accuracy maximization.

2 Active Learning: Prior Work

The fundamental notion of active learning has a long history in machine learning. To our knowledge, the first to discuss it explicitly were [Simon and Lea, 1974] and [Winston, 1975]. Simon and Lea describe how machine learning is different from other types of problem solving, because learning involves the simultaneous search of two spaces: the hypothesis space and the instance space. The results of searching the hypothesis space can affect how the instance space will be searched. Winston discusses how the best examples to select next for learning are "near misses," instances that miss being class members for only a few reasons. Subsequently, theoretical results showed that the number of training data can be reduced substantially if they can be selected carefully [Angluin, 1988; Valiant, 1984]. The term *active learning* was coined later to describe induction where the algorithm controls the selection from a set of potential training examples [Cohn *et al*., 1994].

Input: an initial labeled set *L*, an unlabeled set *UL*, an inducer *I*, a stopping criterion, and an integer *M* specifying the number of actively selected examples in each phase.
While stopping criterion not met
  /* perform next phase: */
  Apply inducer *I* to *L*
  For each example $\{x_i \mid x_i \in UL\}$ compute ES_i, the effectiveness score
  Select a subset *S* of size M from *UL* based on ES_i
  Remove *S* from *UL*, label examples in *S*, and add *S* to *L*
Output: estimator *E* induced with *I* from the final labeled set *L*

Figure 2: Generic Active Learning Algorithm

A generic algorithm for active learning is shown in Figure 2. A learner first is applied to an initial set *L* of labeled examples (usually selected at random or provided by an expert). Subsequently, sets of *M* examples are selected in phases from a set of unlabeled examples *UL*, until some predefined condition is met (e.g., the labeling budget is exhausted). In each phase, each candidate example $x_i \in UL$ is given an effectiveness score ES_i based on its contribution to an objective function, reflecting the estimated magnitude of its contribution to subsequent learning (or simply whether it will or will not contribute). Examples then are ranked by their effectiveness scores and the top *M* examples are selected for labeling. Usually, multiple examples, rather than a single example, are selected at each phase due to computational constraints. Once examples are selected, their labels are obtained (e.g., by querying an expert) before being added to *L*, on which the learner is applied next.

Cohn et al. [Cohn *et al*., 1994] determine ES_i based on identifying what they called the "region of uncertainty," defined such that concepts from the current version space are inconsistent with respect to examples in the region. The region of uncertainty is redetermined at each phase and subsequent examples are selected from this region. The main practical problem with this approach is that the estimation of the uncertainty region becomes increasingly difficult, as the concept becomes more complex. In addition, for complex concepts the region of uncertainty initially may span the entire domain before the concept is well understood, rendering the selection process ineffective. A closely related approach is Query By Committee (QBC)[Seung *et al.*,1992]: classifiers are *sampled* from the version space, and the examples on which they disagree are considered for labeling. However, QBC is a theoretical approach that poses computational and practical constraints. Particularly, it assumes the existence of hypotheses from the version space available for sampling, as well as noise-free data. Several other approaches also address learning for classification. These methods target examples for which predictions of class membership are evenly split (for binary classes) among an ensemble of classifiers, or alternatively examples for which a single probabilistic classifier assigns CPE near 0.5, as indicating class uncertainty. Specifically, Lewis and Gale [Lewis and Gale, 1994] proposed UNCERTAINTY SAMPLING where a probabilistic classifier is employed, and examples whose probabilities of class membership are closest to 0.5 are considered for labeling. Abe and Mamitsuka [Abe and Mamitsuka, 1998] generate a set of classifiers and then select examples for which the classifiers are close to being evenly split. An approach due to Iyengar et al. [Iyengar *et al*., 2000] directly estimates whether a classifier will assign an example to the wrong class. They employ a second classifier to assign classes to unlabeled examples, and examples are considered more informative for learning if estimated as being likely to be *misclassified* by the current ensemble of classifiers. These approaches are designed specifically for maximizing classification accuracy, not for optimizing CPEs or rankings, which are our concern.

The method most closely related to our technique was presented by Cohn et al. [Cohn *et al*., 1996] for statistical learning models. At each phase they compute the expectation of the variance of the model over the example space resulting from adding each candidate example to the training set. Our approach is similar in that it estimates variance, but instead of modeling the variance of the model over the input space, we estimate the "local" variance for each

$x_i \in UL$. The approach of Cohen *et al*. requires knowledge of the underlying domain, as well as the computation in closed form of the learner's variance, a constraint that renders it impracticable for arbitrary models. Our approach can be used for arbitrary models.

3 The Bootstrap-LV Algorithm

BOOTSTRAP-LV actively samples examples from *UL* to learn *class probability estimates* (CPEs). The description we provide here pertains to binary class problems where the set of class labels is $C = \{0,1\}$. As the discussion above indicates, we wish to add to L examples that are likely to improve the available evidence pertaining to poorly understood subspaces of the example space.

Ideally, the most direct indication of the *quality* of the current class probability estimate for example x_i is the discrepancy between the estimated probability and its true probability. However, the true class probability for an instance is not known, nor is its actual class. Therefore we use the "local variance" (LV) to estimate this quality. Local variance refers to the variance in CPE for a particular example. If the estimated LV is high compared to that of other examples, we infer that this example is "difficult" for the learner to estimate given the available data, and is thus more desirable to be selected for learning. Otherwise, if the LV is low, we interpret it as an indication that either the class probability is well learned or, on the contrary, that it will be extremely difficult to improve. We therefore decrease the likelihood of these examples being added to L.

Given that a closed-form computation/estimation of this local variance may not (easily) be obtained, we estimate it empirically. We generate a set of *k bootstrap* subsamples [Efron and Tibshirani, 1993] B_j, $j = 1,...,k$ from L, and apply the inducer *I* to each subsample to generate *k* estimators E_j, $j = 1,...,k$. For each example in *UL* we estimate the variance in CPEs given by the estimators $\{E_j\}$. Each example in *UL* is assigned a weight, which determines its probability of being sampled, and which is proportional to the variance of the CPEs. More specifically, the distribution from which examples are sampled is given by $D_s(x_i) = \frac{\left\{\sum_{j=1}^{k}\left[\left(p_j(x_i) - \overline{p}_i\right)^2\right]\right\} / \overline{p}_{i,\min}}{R}$, where $p_j(x_i)$ denotes the estimated probability an estimator E_j assigns to the event that example x_i belongs to class 0 (the choice of performing the calculation for class 0 is arbitrary, since the variance for both classes is identical and hence the same result for $D_s(x_i)$ is obtained for class 1); $\overline{p}_i$ is the average $\frac{\sum_{j=1}^{k} p_j(x_i)}{k}$; $\overline{p}_{i,\min}$ is the average probability estimation assigned to the minority class by the various estimators, and R is a normalizing factor $R = \sum_{i=1}^{size(UL)}\left\{\sum_{j=1}^{k}\left[\left(p_j(x_i) - \overline{p}_i\right)^2\right]\right\} / \overline{p}_{i,\min}$, so that D_s is a distribution. This is the BOOTSTRAP-LV algorithm, shown in Figure 3.

Algorithm BOOTSTRAP-LV

1 **Input**: an initial labeled set L sampled at random, an unlabeled set *UL*, an inducer *I*, a stopping criterion, and a sample size *M*.

2 for (s=1;until stopping criterion is met; s++)
3 Generate *k* bootstrap subsamples B_j, $j = 1,...,k$ from *L*
4 Apply inducer *I* on each subsample B_j and induce estimator E_j
5 For all examples $\{x_i \mid x_i \in UL\}$ compute
$D_s(x_i) = \frac{\left\{\sum_{j=1}^{k}\left[\left(p_j(x_i) - \overline{p}_i\right)^2\right]\right\} / \overline{p}_{i,\min}}{R}$
6 Sample from the probability distribution D_s, a subset S of *M* examples from *UL* without replacement
7 Remove S from *UL*, label examples in S, and add them to *L*
8 end for
9 **Output:** estimator *E* induced with *I* from *L*

Figure 3: The BOOTSTRAP-LV Algorithm

There is one additional technical point of note. Consider the case where the classes are not represented equally in the training data. When high variance exists in regions of the domain for which the minority class is assigned high probability, it is likely that the region is relatively better understood than regions with *the same variance* but for which the majority class is assigned high probability. In the latter case, the class probability estimation may be exhibiting high variance due simply to lack of representation of the minority class in the training data, and would benefit from oversampling from the respected region. That is why we divide the estimated variance by the average value of the minority-class probability estimates $\overline{p}_{i,\min}$. We determine the minority class once from the initial random sample.

4 Experimental Evaluation

We are interested primarily in comprehensible models, so for these experiments we use decision trees to produce class probability estimates. However, BOOTSTRAP-LV applies to any technique for learning CPEs. Particularly, the underlying probability estimator we use is a probability estimation tree (PET)—an unpruned C4.5 decision tree [Quinlan, 1993] for which the Laplace correction [Cestnik, 1990] is applied at the leaves. The Laplace correction has been shown to improve the CPEs produced by PETs [Bauer and Kohavi, 1998; Provost et al., 1998; Provost & Domingos, 2000].

When evaluating CPE accuracy, if the true underlying class probability distribution were known, an evaluation of an estimator's accuracy could be based on a measure of the actual error in probability estimation. Since the true probabilities of class membership are not known we compare the probabilities assigned by the model induced at each phase with those assigned by a "best" estimator, E_B, as surrogates to the true probabilities. E_B is induced from the entire set of examples ($UL \cup L$), using bagged-PETs, which

have been shown to produce superior probability estimates compared to individual PETs [Bauer and Kohavi, 1998; Provost et al., 1998; Provost & Domingos, 2000]. We compute the mean absolute error (MAE) for an estimator E with respect to E_B's estimation, denoted by *BMAE*. Specifically, $BMAE = \frac{\sum_{i=1}^{N} \left| p_{E_B}(x_i) - p_E(x_i) \right|}{N}$, where $p_{E_B}(x_i)$ is the estimated probability given by E_B; $p_E(x_i)$ is the probability estimated by E, and N is the number of examples examined.

To evaluate its performance, we applied BOOTSTRAP-LV to 20 data sets, 17 from the UCI machine learning repository [Blake *et al*., 1998] and 3 used previously to evaluate rule-learning algorithms [Cohen and Singer, 1999]. Data sets with more than two classes were mapped into two-class problems. We compare the performance of BOOTSTRAP-LV against a method denoted by RANDOM, where estimators are induced with the same inducer and training set size, but for which examples are sampled at random. We show the comparison for different sizes of the labeled set L. In order not have very large sample sizes M for large data sets and very small ones for small data sets, we applied different numbers of phases for different data sets, varying between 10 and 30; at each phase the same number of examples was added to L. Results are averaged over 10 random partitions of the data sets into an initial labeled set, an unlabeled set, and a test set against which the two estimators are evaluated. For control the same partitions were used by both RANDOM and BOOTSTRAP-LV.

The banana curve in Figure 1 above shows the relative performance for the *Car* data set (where *Active Learning* refers to BOOTSTRAP-LV). As shown in Figure 1, the error of the estimator induced with BOOTSTRAP-LV decreases faster initially, exhibiting lower error for fewer examples. This demonstrates that examples actively added to the labeled set are more informative (on average), allowing the inducer to construct a better estimator with fewer examples. For some data sets BOOTSTRAP-LV exhibits even more dramatic results; Figure 4 shows results for the Pendigits data set.

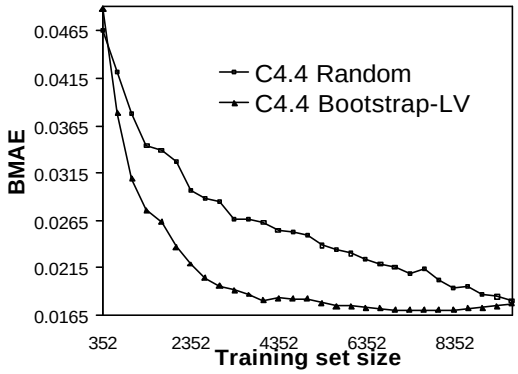


Figure 4: CPE learning curves for the Pendigits data set

BOOTSTRAP-LV achieves its almost minimal level of error at 4500 examples. RANDOM requires more than 9300 examples to obtain this error level. For 5 of the 20 data sets, our approach did not succeed in accelerating learning much or at all, as is shown for the Weather data set in Figure 5. Note, however, that neither curve consistently resides above the other and the two methods' performance is comparable.

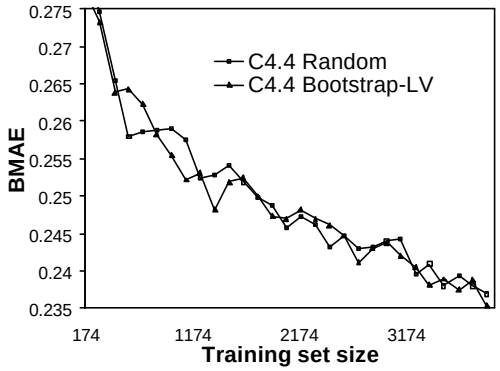


Figure 5: CPE learning curves for the Weather data set

Table 1 presents a summary of our results for all the data sets. The primary motivation for applying active learning techniques is to allow learning with fewer examples. Table 1 provides a set of measures pertaining to the number of examples "gained" using BOOTSTRAP-LV instead of RANDOM. The second column shows the percent of phases in which BOOTSTRAP-LV produced the same level of CPE accuracy with fewer examples than RANDOM (we will call this "phases-gained"). The third and fourth columns show the percentage and number of examples *gained* by applying BOOTSTRAP-LV. The gain is calculated as the difference between the number of examples used by RANDOM and that used by BOOTSTRAP-LV to obtain the same CPE accuracy. The percentage is calculated based on the number of examples used by RANDOM. Because of the natural banana shape even for the ideal case, the performance of estimators induced from any two samples cannot be considerably different at the final phases, thus the averages as well as the phases-gained merely provide an indication of whether BOOTSTRAP-LV produces superior estimations. It is important also to observe the improvement at the "fat" part of the banana (where the benefit of active learning is concentrated). To allow a stable assessment we provide rather than the single best gain, the average of the largest 20% of the gains. Columns 5 and 6 of Table 1 show the average percent and average number (respectively) of examples gained for the top 20% gains. It is important that these figures be viewed in tandem with column 2 (phases-gained), to ensure that there is in fact a banana shape to the graph.

Table 1 also includes summary results pertaining to the error rates achieved by both methods for the same number of examples. Column 7 presents the average error reduction for the 20% of the sampling phases exhibiting the highest error reduction. For some data sets the generalization error for the initial training sets was small and was not

considerably reduced even when the entire data was used for training (e.g., for connect-4, only 34% error reduction was obtained, from 11.7 to 7.7). We therefore also provide, in the last column, the top-20% error gain as a percentage of the reduction required to obtain the minimal error (the latter is referred to in the table as *maximal gain*). In the Adult data set, for instance, BOOTSTRAP-LV exhibited only 6.6% error gain (for the top 20%), but this improvement constitutes 25% of the possible improvement were the entire data set used for training.

	Examples					Error (%)	
Data set	Phases with positive gain (%)	Avg % gained	Avg # gained	Top 20% % gained	Top 20% # gained	Avg top 20% (%)	Avg top 20% (% from maximal gain)
abalone	**92.5**	**34.9**	**574**	**76.9**	**1152**	**10.1**	**64.0**
adult	**96**	**17.8**	**302**	**30.2**	**585**	**6.6**	**25.0**
breast cancer-w	**100**	**23.8**	**44**	**51.6**	**110**	**9.3**	**41.0**
car	**89.6**	**23.3**	**155**	**35.4**	**281**	**31.3**	**53.3**
coding1	**80**	**16.2**	**228**	**47.1**	**475**	**2.5**	**28.9**
connect-4	**100**	**45.5**	**984**	**75.4**	**1939**	**9.5**	**27.5**
contraceptive	**93.7**	**18.4**	**55**	**42.3**	**129**	**5.7**	**31.3**
german*	57.1	5.8	7	46.5	113	5.9	31.0
hypothyroid	**100**	**64.6**	**705**	**69.0**	**1233**	**41.1**	**72.4**
kr-vs-kp	**100**	**18.1**	**37**	**27.1**	**57**	**25.5**	**30.8**
letter-a**	**72.4**	**14.5**	**229**	**24.8**	**529**	**10.4**	**26.0**
letter-vowel	50	2.1	121	12.8	429	3.4	18.0
move1	65	17.2	23	68.4	75	3.9	12.8
ocr1	**93.7**	**24.5**	**83**	**42.9**	**168**	**21.7**	**65.0**
optdigits	**94.4**	**24.5**	**412**	**50.0**	**762**	**32.6**	**47.8**
pendigits	**100**	**61.0**	**3773**	**68.6**	**5352**	**29.9**	**75.6**
sick-euthyroid	**93.1**	**45.2**	**600**	**70.2**	**924**	**26.2**	**58.5**
solar-flare	64.2	13.5	25	41.5	58	6.3	9.9
weather	41.6	-10.4	-46	35.9	438	1.7	20.1
yeast	**75**	**23.6**	**79**	**58.7**	**159**	**4.9**	**30.8**

* German credit database

** letter-recognition, letter a

Table 1: Improvement in examples needed and improvement in error using BOOTSTRAP-LV

Since not all plots can be presented due to space constraints, we tried to express in the table various performance measures that would provide a comprehensive perspective. To assess BOOTSTRAP-LV's superiority we apply the combination of the following: phases-gained should be above 60%; both the average example and error gains should be positive, and the top-20% error reduction from the maximal gain should be 25% or higher. If phases-gained is between 40% and 60% we consider the methods to be comparable, and when it is below 40% we consider BOOTSTRAP-LV to be inferior. As can be seen in Table 1 (in bold), in 15 out of the 20 data sets BOOTSTRAP-LV exhibited superior performance. Particularly, in all but one phases-gained is 75% or above. In 13 of those, more than 30% of the examples were saved (for the top 20%), and in 9 data sets our method used less than 50% of the number of examples required for RANDOM to achieve the same level of accuracy. For the Sick-euthyroid data set, for instance, BOOTSTRAP-LV gradually improves until it requires fewer than 30% of the examples required by RANDOM to obtain the same level of accuracy. These results pertain to the top-20% improvement, so the maximal gain can be much higher.

For a single data set (Weather) BOOTSTRAP-LV exhibited a negative average examples gain. However, phases-gained, showing that BOOTSTRAP-LV uses fewer examples in 41% of phases examined, and Figure 5, both indicate that the two methods indeed exhibit comparable learning curves for this data set.

The measures pertaining to the number of examples gained and the error gain complement each other and may provide interesting insight. For instance, the number of examples gained can help evaluate the "difficulty" in error reduction in terms of the number of examples required by RANDOM to obtain such reduction. For example, although the average top-20% error gain for Connect-4 was less than 10%, Table 1 shows that it required RANDOM 984 additional examples on average to obtain the same improvement. A single data set, Letter-vowel, exhibited a negative average error gain. However, phases-gained is exactly 50%, indicating that RANDOM indeed does not exhibit superior performance overall. Both methods have similar learning curves.

We also assessed both methods with two alternatives to BMAE: the mean squared error measure proposed by Bauer and Kohavi [1998], as well as the area under the ROC curve [Bradley 1997] which specifically evaluates ranking accuracy. The results for these measures agree with those obtained with BMAE. For example, Bootstrap-LV generally leads to fatter ROC curves with fewer examples.

For those data sets in which BOOTSTRAP-LV exhibits insignificant or no improvement at all, training examples chosen at random seem to contribute to error reduction at an almost constant rate. Their learning curves have an atypical shape, as shown for the Weather data set in Figure 5, where additional examples bring an almost constant reduction in error rather than the expected decreasing marginal error reduction. This may indicate that it is easy to obtain good examples for learning, and any additional example contributes to error reduction equally, regardless of what or how many examples have been already used for training. Thus intelligent selection of learning examples is less likely to improve learning significantly.

5 Additional Experiments

Tree-based models offer a comprehensible structure that is important in many decision-making contexts. However, they often do not provide the best probability estimates. In order to assess BOOTSTRAP-LV 's performance on a better CPE learner, we experimented with bagged-PETs, which are not comprehensible models, but have been shown to produce markedly superior CPEs [Bauer and Kohavi, 1998; Provost et al., 1998; Provost & Domingos, 2000].

The results for the bagged-PETs model agree with those obtained for individual PETs. Particularly, for 15 of the data sets BOOTSTRAP-LV exhibited phases-gained of more than 65% (in 13 of those phases-gained is more than 75%). The average top-20% example gain was 25% or higher in 11 of those data sets. Only in two data sets is phases-gained less than 50%. Figure 6 shows a comparison between BOOTSTRAP-LV and RANDOM for individual PETs and for bagged-PETs. As expected, the overall error exhibited by

the bagged-PETs is lower than for the PET, and for both models BOOTSTRAP-LV achieves its lowest error with considerably fewer examples than are required for RANDOM.

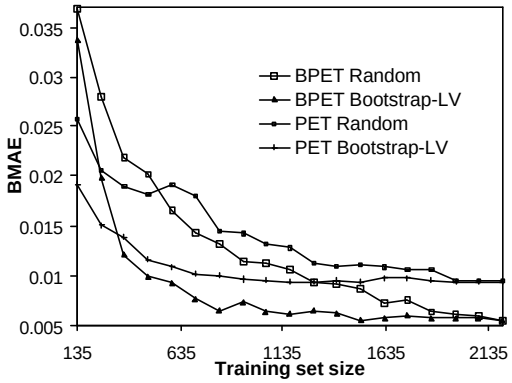


Figure 6: CPE learning curves for the Hypothyroid data set

Described above, UNCERTAINTY SAMPLING [Lewis and Gale, 1994] was proposed for binary text classification. However, it too samples examples that are not well understood by the model. Since it was shown to improve a model's classification accuracy, it may improve the model's CPE as well. It therefore is interesting to compare the improvements exhibited by BOOTSTRAP-LV against UNCERTAINTY SAMPLING. We present a summary of the comparison results in Table 2, where all the measures are the same as in Table 1, except that the baseline comparison is UNCERTAINTY SAMPLING rather than RANDOM.

	Examples					Error (%)	
Data set	Phases with positive gain (%)	Avg % gained	Avg # gained	Top 20% % gained	Top 20% # gained	Avg top 20% (%)	Avg top 20% (% from maximal gain)
abalone	50.00	17.63	102	61.09	801	14.11	57.57
adult	**69.23**	**9.56**	**69**	**35.03**	**284**	**11.13**	**27.18**
breast cancer-w	55.56	10.90	15	49.37	144	20.20	43.91
car	**62.50**	**9.95**	**6**	**50.46**	**68**	**36.30**	**43.26**
coding1	**93.75**	**31.77**	**686**	**63.25**	**1027**	**6.74**	**49.26**
connect-4	**89.47**	**43.89**	**1958**	**85.52**	**3230**	**54.02**	**82.91**
contraceptive	50.00	11.76	21	54.87	126	10.01	29.13
German	**81.25**	**24.74**	**69**	**48.14**	**146**	**8.12**	**37.63**
hypothyroid	**71.43**	**17.10**	**85**	**62.30**	**307**	**62.72**	**77.74**
kr-vs-kp	**94.74**	**33.90**	**90**	**57.71**	**144**	**60.43**	**64.07**
letter-a	**85.00**	**15.50**	**395**	**44.34**	**771**	**21.29**	**30.65**
letter-vowel	**100.00**	**63.80**	**11463**	**81.27**	**14210**	**44.97**	**43.41**
move1	**100.00**	**39.96**	**194**	**62.89**	**247**	**16.29**	**36.26**
ocr1	**100.00**	**35.86**	**146**	**51.90**	**256**	**34.30**	**61.75**
optdigits	**100.00**	**26.08**	**570**	**44.13**	**1359**	**34.91**	**58.16**
pendigits	**95.00**	**27.45**	**996**	**60.85**	**1636**	**38.30**	**58.03**
sick-euthyroid	**100.00**	**59.13**	**1093**	**84.12**	**1692**	**40.51**	**64.49**
solar-flare	0.00	-16.66	-69	-2.98	-17	-1.64	-6.54
weather	56.25	6.32	3	35.06	351	1.98	24.74
yeast	53.33	7.74	3	40.38	121	6.03	28.88

Table 2: Summary results of BOOTSTRAP-LV versus UNCERTAINTY SAMPLING (CPE)

BOOTSTRAP-LV exhibits markedly superior performance compared to UNCERTAINTY SAMPLING. Particularly, BOOTSTRAP-LV is superior in 14 of the data sets, and in 5 data sets the methods exhibit comparable performance, where phases-gained for BOOTSTRAP-LV between 50% and 60%. UNCERTAINTY SAMPLING exhibits superior performance in one data set, Solar-Flare, for which it consistently produces better probability estimations.

In 9 out of the 14 data sets in which BOOTSTRAP-LV was superior, the average top error reduction was more than 30%. These results demonstrate that BOOTSTRAP-LV has a solid advantage when compared to UNCERTAINTY SAMPLING for class probability estimation. Moreover, for several data sets UNCERTAINTY SAMPLING'S performance was inferior to that of RANDOM. It is important to emphasize once again that indeed UNCERTAINTY SAMPLING was not designed to improve class probability estimation, but rather to improve classification accuracy.

We also compared the performance of UNCERTAINTY SAMPLING against BOOTSTRAP-LV for improving classification accuracy. Since BOOTSTRAP-LV was found to improve CPEs, a similar effect may be obtained for classification accuracy, but not necessarily: BOOTSTRAP-LV may select examples to improve class probability estimation even when the estimated decision boundary required for classification is already well understood, thereby "wasting" examples that do not improve classification accuracy.

Our results for classification accuracy show that in 11 data sets BOOTSTRAP-LV exhibited superior performance for accuracy maximization. UNCERTAINTY SAMPLING was superior in 7 data sets and the methods exhibited comparable performance for the remaining two. These results indicate that although BOOTSTRAP-LV is not uniformly superior to UNCERTAINTY SAMPLING for classification tasks, it should be considered a viable alternative—it often yields much better performance. Interestingly, in most cases where BOOTSTRAP-LV does not dominate, it performs better in the initial phases, whereas UNCERTAINTY SAMPLING surpasses BOOTSTRAP-LV in later phases. This phenomenon is demonstrated in Figure 7 for the Breast-Cancer data set.

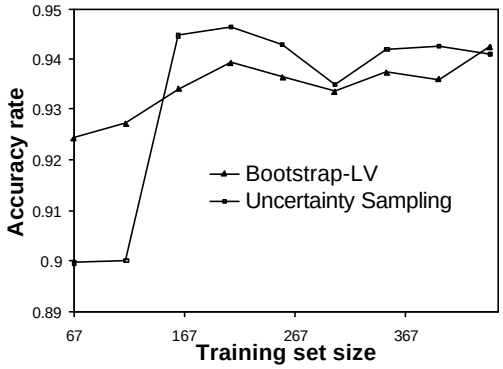


Figure 7: Classification accuracy rate for Breast-Cancer

Recall that UNCERTAINTY SAMPLING uses the CPEs to determine the potential contribution of an example for learning. Therefore, its performance will be sensitive to CPE accuracy. Poor CPEs produced in the initial phases undermine the data selections by UNCERTAINTY SAMPLING. On the other hand, in later phases, more accurate probability esti-

mations allow the selection process to focus in on the decision boundary. BOOTSTRAP-LV, on the contrary, focuses early on improving the CPEs, and therefore performs well even very early on the learning curve; however, later on it indeed "wastes" examples to improve CPE.

In light of this behavior, a better strategy for actively improving classification accuracy may be a hybrid approach: BOOTSTRAP-LV is applied in initial phases and UNCERTAINTY SAMPLING later. "When to switch?" is an open question.

6 Conclusions and Limitations

We introduced a new technique for active learning. BOOTSTRAP-LV was designed to use fewer labeled training data to produce better class probability estimates from fewer labeled data. We showed empirically that it does this remarkably well. We also showed that BOOTSTRAP-LV is competitive with UNCERTAINTY SAMPLING even for accuracy maximization. Inspecting these last results also suggests a hybrid strategy that may be even more effective than either technique alone.

BOOTSTRAP-LV was designed to identify particularly informative examples to use for training in order to economize on labeling costs to obtain higher CPE accuracy. It does not address computational concerns, as do Lewis and Catlett [Lewis and Catlett, 1994]. Indeed BOOTSTRAP-LV is a computationally intensive approach, because of the need to induce at each phase multiple models from a set of bootstrap samples. Yet, because of the typical shape of the learning curve, beyond a certain training set size the marginal error reduction is insignificant, whether active learning or random sampling is employed. Thus, intelligent selection of examples for learning is only critical in the early part of the curve, where a relatively small number of examples are used for training. Therefore, as long as the number of training examples remains relatively small—multiple model inductions from these samples do not constitute a considerable computational toll. Moreover, BOOTSTRAP-LV provides an appropriate solution whenever labeling costs are more important than computational costs, for example, when the primary concern is to obtain accurate CPE or ranking with minimal costly labeling.

Acknowledgments

We thank Vijay Iyengar for helpful comments and IBM for a Faculty Partnership Award.

References

[Abe and Mamitsuka, 1998] Abe, N. and Mamitsuka, H. Query Learning Strategies using Boosting and Bagging. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 1-9.

[Angluin, 1988] Angluin, D. Queries and concept learning. *Machine Learning*, 2:319-342, 1988.

[Bauer and Kohavi, 1998] Bauer, E., Kohavi, R. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, 36, 105-142 (1998).

[Blake *et al.*, 1998] Blake, C.L. & Merz, C.J. UCI Repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science, 1998 [http://www.ics.uci.edu/~mlearn/MLRepository.html].

[Bradley, 1997] Bradley, A. P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7), 1145-1159, 1997.

[Cestnik, 1990] Cestnik, B. Estimating probabilities: A crucial task in machine learning. *In Proceedings of the Ninth European Conference on Artificial Intelligence*, 147-149, Sweden, 1990.

[Cohn et al., 1994] Cohn, D., Atlas, L. and Ladner, R. Improved generalization with active learning. *Machine Learning*, 15:201-221, 1994.

[Cohn *et al.*, 1996] Cohn, D., Ghahramani, Z., and Jordan M. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129-145, 1996.

[Cohen and Singer, 1999] Cohen, W. W. and Singer, Y. A simple, fast, and effective rule learner. In AAAI-99, 335-342, 1999.

[Efron and Tibshirani, 1993] Efron, B. and Tibshirani, R. *An introduction to the Bootstrap*, Chapman and Hall, 1997.

[Iyengar *et al.*, 2000] Iyengar, V. S., Apte, C., and Zhang T. Active Learning using Adaptive Resampling. *In Sixth ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*. 92-98.

[Lewis and Gale, 1994] Lewis, D. and Gale, W. A. A sequential algorithm for training text classifiers. In *ACM-SIGIR-94*, 3-12.

[Lewis and Catlett, 1994] Lewis, D. D., and Catlett, J. Heterogeneous uncertainty sampling. In *Proceedings of the Eleventh International Conference on Machine Learning*, 148-156, 1994.

[Provost *et al.*, 1998] Provost, F.; Fawcett, T.; and Kohavi, R. The case against accuracy estimation for comparing classifiers. In *Proc. of the Intl. Conf. on Machine Learning*, 445-453, 1998.

[Provost and Domingos, 2000] Provost, F. and Domingos P. Well-trained PETs: Improving Probability Estimation Trees. *CeDER Working Paper #IS-00-04, Stern School of Business, NYU*.

[Quinlan, 1993] Quinlan, J. R.. *C4.5: Programs for machine learning*. Morgan Kaufman, San Mateo, California, 1993.

[Seung *et al.*, 1992] H. S. Seung, M. Opper, and H. Smopolinsky. Query by committee. *In Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, 287-294, 1992.

[Simon and Lea, 1974] Herbert A. Simon and Glenn Lea, Problem solving and rule induction: A unified view. In L.W. Gregg (ed.), Knowledge and cognitiom. Chap. 5. Potomac, MD: Erlbaum, 1974.

[Turney, 2000] Turney, P.D. Types of cost in inductive concept learning, *Workshop on Cost-Sensitive Learning at ICML-2000*, Stanford University, California, 15-21.

[Valiant, 1984] Valiant L. G. A theory of the learnable. *Communications of the ACM*, 27:1134-1142, 1984.

[Winston, 1975]. Winston, P. H. Learning structural descriptions from examples. In *"The Psychology of Computer Vision"*, P. H. Winston (ed.), McGraw-Hill, New York, 1975.

[Zadrozny and Elkan, 2001] Zadrozny B. and Elkan C. Learning and making decisions when costs and probabilities are both unknown. Technical Report No.CS2001-0664, Dept. of Computer Science and Engineering, UC San Diego, January 2001.

MACHINE LEARNING AND DATA MINING

MACHINE LEARNING

Learning on the Phase Transition Edge

Alessandro Serra, Attilio Giordana and Lorenza Saitta
Dipartimento di Scienze e Tecnologie Avanzate,
Università del Piemonte Orientale,
Corso Borsalino 54, 15100, Alessandria, Italy

Abstract

A previous research has shown that most learning strategies fail to learn relational concepts when descriptions involving more than three variables are required. The reason resides in the emergence of a phase transition in the *covering test*. After an in depth analysis of this aspect, this paper proposes an alternative learning strategy, combining a Monte Carlo stochastic search with local deterministic search. This approach offers two main benefits: on the one hand, substantial advantages over more traditional search algorithms, in terms of increased learning ability, and, on the other, the possibility of an a-priori estimation of the cost for solving a learning problem, under specific assumptions about the target concept.

1 Introduction

In a recent paper Giordana et al. [Giordana *et al.*, 2000] have shown that relational learning becomes very hard when the target concept requires descriptions involving more than three variables. The reason is related to the presence of a phase transition in the covering test [Prosser, 1996; Giordana and Saitta, 2000], i.e., an abrupt change in the probability that an inductive hypothesis covers a given example, when the hypothesis and the example sizes reach some critical values. Moreover, any top-down learner will search for discriminant hypotheses in the phase transition region [Giordana *et al.*, 2000; Giordana and Saitta, 2000], and, finally, heuristics commonly used to guide top-down relational learning [Quinlan, 1990; Botta and Giordana, 1993] become useful only in the same region. The consequence is that the top-down induction process is blind in its first steps, so that the path to the correct concept definition is very easily lost.

This paper offers two major contributions. First, the results reported by [Giordana *et al.*, 2000] are extended, providing a more thorough analysis of the behavior of a top-down learner. A formal estimate of the "difficulty" of a learning problem is proposed, in terms of the density of sub-formulas of the target concept in the hypothesis space. Moreover, an estimate of the probability of detecting one of these is also given, as a function of the target concept location with respect to the phase transition.

Second, an induction algorithm, combining a Monte Carlo stochastic search [Brassard and Bratley, 1988] with local deterministic search, is proposed to (partially) avoid the pitfall that causes top-down search to fail. The new algorithm directly jumps into a region of the hypothesis space where the information gain heuristics has good chance of being successful, continuing its search exploiting a classical hill climbing strategy. The complexity of this algorithm is analyzed in a probabilistic framework, and it is proposed as a measure of the difficulty of the induction task. Finally, the algorithm has been experimentally evaluated on the set of hard induction problems provided by [Giordana *et al.*, 2000], and it shows a good agreement with the theoretical estimates.

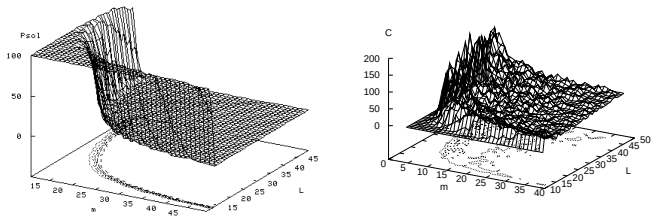
2 Covering Test in Relational Learning

A key step in Machine Learning is the *covering test*, i.e., checking whether an example e belonging to a learning set E verifies a hypothesis φ [Mitchell, 1997]. In relational learning e is usually represented as a conjunction of ground literals $e(a_1, \ldots, a_r)$, where the a_i's are constants belonging to a given set [Muggleton and De Raedt, 1994], and φ is a conjunction of non-ground literals $\varphi(x_1, .., x_n)$ containing the variables $x_1, .., x_n$. The covering test checks whether a First Order Logic formula φ has at least one model in the universe e. To this aim, a substitution θ of the variables $x_1, .., x_n$ with constants $a_1, \ldots, a_r$ that satisfies φ is searched for. [1].

This search process shows a phase transition for critical values of the number L of constants a_i's in the universe, and the number m of literals in the hypothesis φ [Hogg *et al.*, 1996; Prosser, 1996; Botta *et al.*, 1999]. The graph in the (m, L) plane of the probability that a model of φ exists in e is reported in Figure 1(a), for a set of covering problems generated according to a random model described in [Botta *et al.*, 1999]. In the following, we will refer to "phase transition" as to the locus of the points (m_c, L_c) for which the probability that a model exists is $P_{sol} = 0.5$. The phase transition is located in correspondence of the "waterfall" in Figure 1(a). To the left of the phase transition (in the YES-region, i.e., the high plateau in Figure 1(a)) almost any example verifies any hypothesis. To the right of the phase tran-

[1] Verifying φ on e is a Constraint Satisfaction Problem, which exhibits a typical phase transition [Williams and Hogg, 1994; Smith and Dyer, 1996; Prosser, 1996].

sition (in the low plateau, i.e., the NO-region) almost no example verifies any hypothesis. The steep step corresponds to the "mushy" region, where the probability of a model existing drops from 0.99 to 0.01.In this region, the complexity of the search reaches its maximum, as reported in Figure 1(b).More details can be found in [Giordana *et al.*, 2000; Botta *et al.*, 1999].



(a) (b)

Figure 1: Example of phase transition in the covering test problem. All hypotheses have four variables and contain only binary predicates. Each predicate is associated to a table with 100 tuples in each example. (a) Probability that a model exists for 4 variables. (b) Corresponding complexity peak for the search of a model.

If the number of variables in the formulas changes, the location of the phase transition moves, as described in Figure 2.

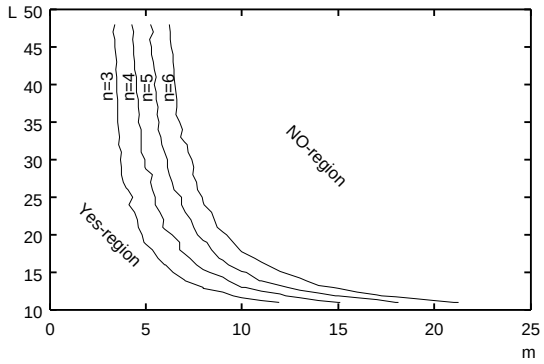


Figure 2: Dependency of the phase transition location upon the number of variables in a hypothesis.

3 Top-Down Induction

Let us consider a learning set $\mathcal{E}$ and a conjunctive concept ω. Every pair (e, ω), with $e \in \mathcal{E}$, represents a covering test (m_ω, L_e) in the (m, L) plane. A learning problem corresponds then to a cloud of points [Giordana *et al.*, 2000]. In the special case that the number of constants in all examples is the same, the entire learning problem $(m_\omega, L_\mathcal{E})$ corresponds to a single point. In a similar way, any inductive hypothesis φ, built up by a learner in the attempt of finding ω, or at least a good approximation $\hat{\omega}$ of it, can be represented as a cloud of points (or a single point) in the (m, L) plane. Then, an induction process that proceeds by generating a sequence of hypotheses will describe a family of lines (or a single line) parallel to the m axis.

In order to investigate how the presence of a phase transition in the (m, L) plane may affect the behavior of a relational learner, an extensive experimentation has been done using a large set of artificially generated learning problems [2]. The results have been reported and discussed in [Giordana *et al.*, 2000]. Here experimental setting and the findings are recalled, in order to state the base line for the work described in this paper.

The artificial learning problem set has been generated according to the following procedure. Given n and N, let (m, L) be a point in the plane (m, L). A set of 451 points has been sampled in the plane (m, L). Then, for each point, a learning problem $\Pi_{m,L}$ has been built up from the 4-tuple $(n = 4, N = 100, m, L)$. A target concept ω with m literals and n variables has been built up, and then a training set $\mathcal{E}_L$ and a test set $\mathcal{E}_T$ have been generated. Let Λ be a set of L constants; every example is a collection of m relational tables of size N obtained by sampling the binary table $\Lambda \times \Lambda$. In order to generate balanced training and test sets in each point of the (m, L) plane, the random generation of the examples has been modified in order to obtain examples with models also in the NO-region, and examples without models also in the YES region (see [Giordana *et al.*, 2000] for more details). The result has been the generation of training and test sets, $\mathcal{E}_L$ and $\mathcal{E}_T$, each one with 100 positive and 100 negative examples in each (m, L) point.

The experimentation has been done primarily using FOIL [Quinlan, 1990]. A problem has been considered solved when the concept description found on $\mathcal{E}_L$ was at least 80% accurate on $\mathcal{E}_T$. However, other learners, such as SMART+ [Botta and Giordana, 1993] and G-NET [Anglano *et al.*, 1998], have also been tried for comparison, reporting a strong agreement with FOIL both for the positive and for the negative cases. The results can be summarized as in the following. First, all tried learning strategies always end up generating concept descriptions lying on, or very close to, the edge of the phase transition. As discussed by [Giordana and Saitta, 2000], this phenomenon systematically occurs both in artificial and in real learning problems, as soon as the generated hypotheses involve more than three variables.

Second, a large "blind spot" located across the mushy region has been found: when the target concept lies there, no tried learner has been able to find any good generalization of the target concept (see Figure 3). Quite surprisingly, learning problems become again solvable by top-down induction when they are located in the NO-region, to the far right of the phase transition.

Finally, the computational complexity of matching inductive hypotheses lying in the mushy region may render the whole induction process infeasible. A possible way out has been proposed by [Giordana and Saitta, 2000], who suggested to use a stochastic algorithm for on-line estimation of the

[2] All the artificial learning problems defined in this experimental setting are available at: http://www.mfn.unipmn.it/ãttilio/PROJECTS/phase- trans.html.

complexity, at the possible expenses of correctness.

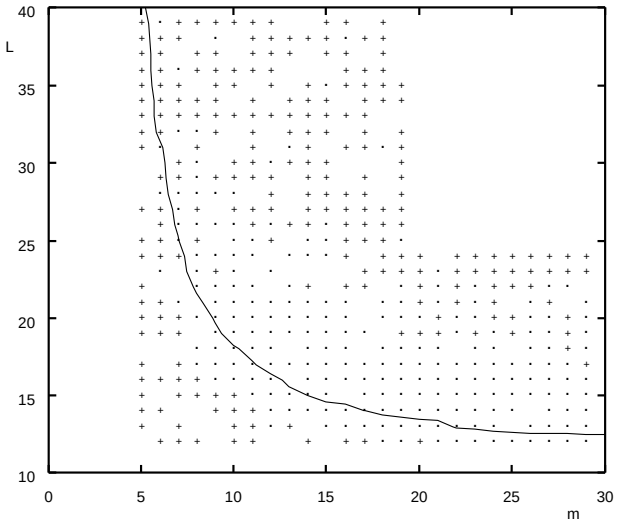


Figure 3: Relational learning with FOIL: *Failure region* (legend ".") and *success region* (legend "+"), for $n = 4$ and $N = 100$. The contour plot corresponds to the value $P_{sol} = 0.5$ of the probability that randomly generated covering test is positive.

4 Why Top-Down Induction is Misled

Before possibly designing new heuristics, it is necessary to clearly understand why FOIL and the other learners fail in the blind spot. A preliminary analysis showed that heuristics relying on the number of models, hypotheses have on the examples [Quinlan, 1990; Botta and Giordana, 1993; Rissanen, 1978], are not reliable in the YES-region, because both wrong and correct generalizations of the target concept have there similar numbers of models both on the positive and on the negative examples. In support to this claim, Figure 4(a) plots the average number μ of models a random hypothesis has as a function of m, for $L = 18$. It is clear that an increase by 1 in the number n of variables induces a large increase of μ and an even larger one on μ's standard deviation σ. Actually, when $n = 4$ the standard deviation is larger than the number of positive examples in the learning set. Then, even if we assume that generalizations of the target concept ω have at least one model more on the positive examples that on the negative ones, these generalizations cannot be distinguished from a purely random hypothesis. On the contrary, for hypotheses with 2 or 3 variables only, μ is much smaller, and we may expect that, provided that a correct generalization $\hat{\omega}$ of ω with only two or three variables exists, it should be far easier to find it than any other one having four variables. This conjecture agrees with the fact that many solutions actually generated by FOIL have only three variables, even though the target concept is described by a formula with four variables.

A more detailed analysis is reported in Figure 5, where the line corresponds to the phase transition for $n = 4$. In Figure 5(a) the "+" denote the locations of a set of target concepts ω, whose description contains four variables. Figure $5(a')$ shows the location of the solutions found by FOIL. Most of

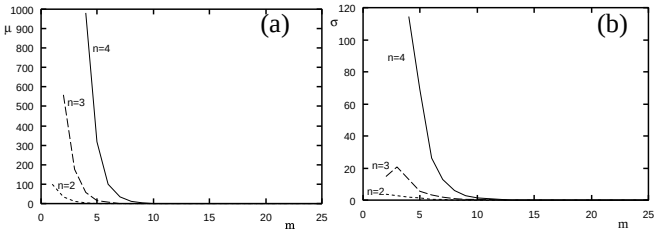


Figure 4: Number μ of models of a random hypothesis versus m, for various values of n. (a) μ for $n = 2$, $n = 3$ and $n = 4$. (b) Standard deviation of μ.

these solutions are generalizations of ω with four variables, because no acceptable generalizations with three or less variables could be found. On the contrary, Figure $5(b)$ shows the locations of a set of target concepts whose description still contains four variables, but for which FOIL was able to find generalizations with no more than three variables (reported in Figure $5(b')$). Problems in Figure 5(b) are then easier to solve than problems in Figure 5(a): this result corresponds to the fact that problems in Figure 5(a) are closer to the phase transition than problems in Figure 5(b). We observe that the probability of a problem being correctly approximated by a formula with only three variables increases with the number of literals in ω. In fact, when ω contains more literals, the number of its subformulas that are correct generalizations of ω increases as well; then, we may expect that at least some of them contains a number of variables smaller than ω's. Moreover, the difficulty of a learning problem increases when L decreases, whereas the critical value m_c increases. When $L < 20$ the only solutions found have three variables. In conclusion, most learning problems in the NO-region have been solved only because approximations with only three variables existed. When more than three variables are required, m_c becomes larger, and available heuristics for top-down search are unable to grow valid inductive hypotheses starting from the YES-region. On the other hand, we could not find any bottom-up learning algorithm capable to cope with the complexity of working directly inside the NO-region.

5 A Stochastic Approach

In the absence of reliable heuristics, stochastic search may be a valid alternative, especially if combined with deterministic search. An example of an effective combination of a Monte Carlo search with deterministic search in the n-Queens problem can be found in [Brassard and Bratley, 1988]. The algorithm we propose is based on a two-step strategy. The first step creates a hypothesis φ_γ with a complexity (number of literals) γ sufficient to reach the border between the YES-region and the mushy region; φ_γ is the result of random sampling of the hypothesis space. The second step consists of a general to specific search, starting from φ_γ according to a hill-climbing strategy guided by the information gain heuristics.

Algorithm SFind
Let $\hat{\omega}_{cur} = \emptyset$
while halt condition does not hold **do**

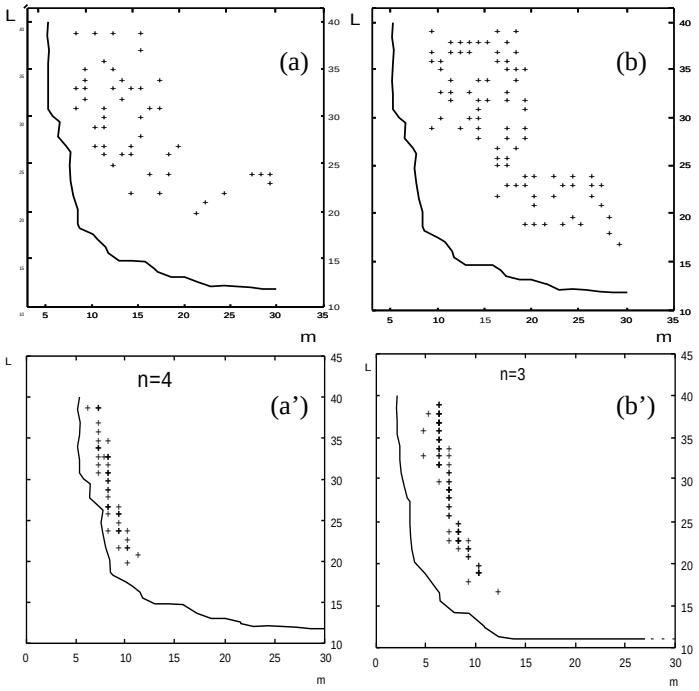


Figure 5: Solutions generated by FOIL for problems in the NO region. (a) Problems solved with four variables; (b) problems solved with hypothesis containing only three variables; (a') and (b') location in the (m, L) plane of the solutions generated by FOIL corresponding to the problems in Figura (a) and (b), respectively.

1. Randomly generate a hypothesis φ_γ close to the mushy region
2. Make φ_γ more specific by following a hill-climbing strategy guided by the information gain. Let $\hat{\omega}_t$ be the locally best hypothesis found in trial t.
3. **if** $\hat{\omega}_t$ is better than $\hat{\omega}_{cur}$ **then** replace $\hat{\omega}_{cur}$ with $\hat{\omega}_t$.

end

If the target concept ω has a conjunctive description in the hypothesis space H, the Monte Carlo algorithm $SFind$ is very likely, in the long run, to find ω, or at least a good generalization $\hat{\omega}$ of it, correct on the learning set $\mathcal{E}$. Now, we want to answer the following question: assuming that the complexity m_ω and the number of variables n_ω in ω are known, what is the complexity we would expect from Algorithm $SFind$? As "complexity" of the algorithm we mean the minimum number T_α of trials necessary to reach a probability (1 - α) of finding $\hat{\omega}$ (or ω, as a specially lucky case). If we choose $\gamma = m_c - 1$, there is a strong experimental evidence that a hill-climbing search, guided by the information gain, almost surely will find $\hat{\omega}$. This happens because the search explores the region where the information gain becomes reliable.

Let P_G be the probability that φ_γ is a sub-formula of ω, then the probability $P_{\hat{\omega}}{}^{(t)}$ of finding $\hat{\omega}$ in no more than t steps is given by the expression:

$$P_{\hat{\omega}}{}^{(t)} = 1 - (1 - P_G)^t \tag{1}$$

Given m_ω and n_ω, the probability P_G can be estimated, for a generic number τ of literals, as the ratio $P_G = N_S(m_\omega, n_\omega, \tau, v)/N_\varphi(m_\omega, \tau, v)$, where $N_S(m_\omega, n_\omega, \tau, v)$ is the number of sub-formulas of ω with τ literals and v variables ($2 \leq v \leq n$), and $N_\varphi(m_\omega, \tau, v)$ is the total number of formulas in the hypothesis space with τ literals and v variables. An analytical expression for $N_S(m_\omega, n_\omega, \tau, v)$ and $N_\varphi(m_\omega, \tau, v)$ has been given in [Serra, 2000], under the assumption that all predicates are binary. Using such expressions, the contour plot of Figure 6 have been obtained for the ratio $R(m_\omega, n_\omega, \tau, v) = N_S(m_\omega, n_\omega, \tau, v)/N_\varphi(m_\omega, \tau, v)$ setting $n_\omega = 4$, $v = 3$ and $v = 4$. It is immediate to verify that $R(m_\omega, n_\omega, \tau, v)$ decreases exponentially with τ, while it remains constant with respect to m, after an initial exponential decay for small values of m. The value P_G can be obtained from Figure 6 by setting $\tau = \gamma$. As $\gamma = m_c - 1$ depends upon L (see Figure 1), P_G depends upon L_c, too. The plots of Figure 7 report the dependency of P_G upon L_c for in the case of $v = 3$ (dotted line), and in the case of $v = 4$ (continuous line). For $n = 3$ the phase transition occurs for smaller values of m_c than for $n = 4$. Consequently, the values of P_G for $v = 3$ are order of magnitude larger than for $v = 4$. This explain why learning a concept with three variables only is much easier than learning a concept with four variables, as it has been stated in the previous section. We

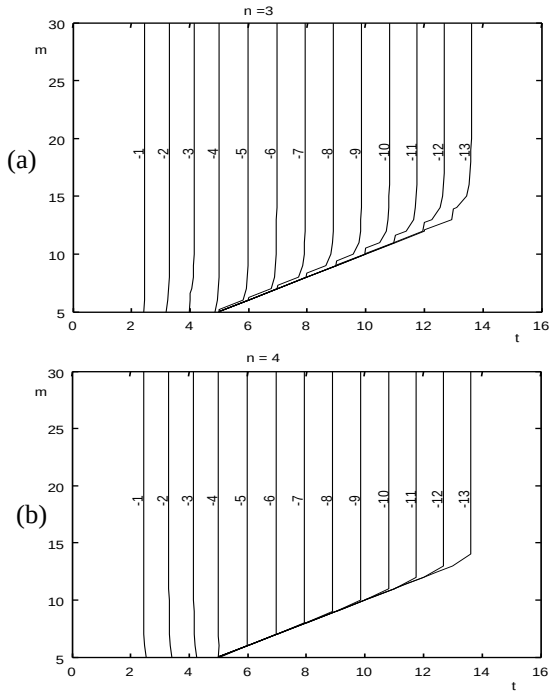


Figure 6: Contour plot of the average proportion of sub-formulas of a target concept ω as a function of m and τ, for $n = 3$ and $n = 4$. (the parameters on the curves are the exponent of 10)

notice that the value L_c is known when the learning examples are given, because L_c is the average number of constants occurring in them.

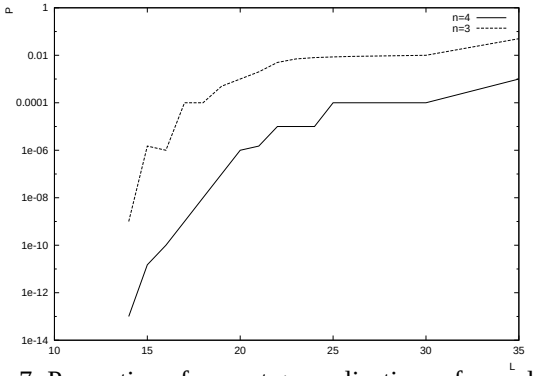


Figure 7: Proportion of correct generalizations of ω evaluated on the edge of the phase transition, i.e., when the probability of solution is $P_{sol} = 0.9$, versus L_c.

6 Improving the Stochastic Search Algorithm

Expression (1) and the graphs in Figure 7 allow one to estimate the number T_α of trials required to reach a confidence (1-α) that Algorithm $SFind$ has found an approximation $\hat{\omega}$, provided that such an approximation exists. Then, the search can be planned as follows:

1. Assume an initial value n_ω for the number of variables in ω.
2. Estimate the number T_α of trials necessary to reach the confidence level $(1-\alpha)$. If T_α is too high, stop, otherwise go to the next step.
3. Run $SFind$ for T_α trials.
4. If the result returned by $SFind$ is acceptable, stop, otherwise increase n_ω by 1 and go to step 2.

We notice that the cost for generating and evaluating a hypothesis in the initial random sampling phase is much lower than the cost for the subsequent hill-climbing search. More specifically, the number of hypotheses to be generated and evaluated for the hill-climbing step can be estimated by:

$$O((1+\nu)mn(n-1)) \tag{2}$$

being ν an integer that experimentally has been observed to range from 0 to 3. In expression (2) the term $(1+\nu)$ is an estimate of the number of literals to be added to a hypothesis φ_γ in order to cross the phase transition, whereas the term $mn(n-1)$ estimates the number of alternative specializations to be considered at each hill-climbing step.

Then, the number of hypotheses to be evaluated in the hill-climbing phase is from one to two orders of magnitude larger than the number of stochastic trials.

A second point worth noticing is that, assuming that the information gain heuristics is reliable for any hypothesis $\varphi_\gamma \wedge \psi$, it is also likely that φ_γ itself be scored higher than the average when it is a sub-formula of ω.

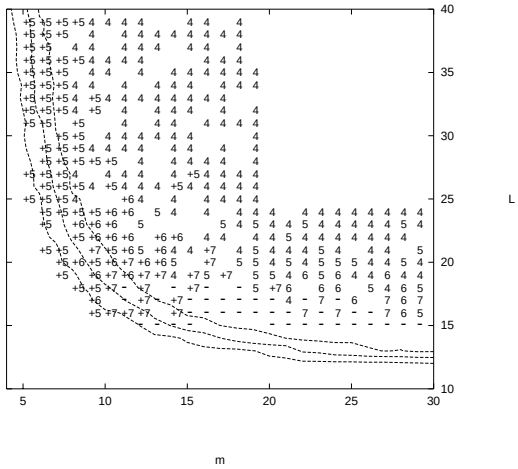


Figure 8: Results obtained by Algorithm T^4. The numbers denote the minimum value that was necessary to assume for γ in order to solve the learning problem. When the number is prefixed by "+", it means that $n = 4$ has been assumed, otherwise $n = 3$. Symbol "." means that the problem has not been solved.

On the basis of the previous considerations we introduce a new algorithm, which can be more effective in practice than Algorithm $SFind$. Let T_α be the number of trials estimated by (1) in order to reach a confidence $(1-\alpha)$ on the output of algorithm $SFind$. Let moreover K $(1 \leq K \leq T_\alpha)$ be a user-defined parameter.

Algorithm T^4

1. Create a set Φ of T_α hypotheses with complexity γ.
2. Rank hypotheses in Φ according to their information gain with respect to the formula $True$.
3. Starting from the top ranked hypothesis, apply the hill-climbing specialization step to the K best ranked hypotheses.
4. Return the best description $\hat{\omega}$.

Algorithm T^4 tries to limit the number of hill-climbing steps to the hypotheses looking more promising, reducing thus the computational complexity. Of course, parameter K is an arbitrary choice, and the confidence level $(1-\alpha)$ is guaranteed to be reached only when $K = T_\alpha$. In practice, we have observed that using values of K relatively small ($K = 1\%$ of T_α) T^4 tends to produce the same results as for $T_\alpha = K$.

Algorithm T^4 has been tested on the set of learning problems, reported in Figure 3, trying to solve the problems laying in the NO-region following the strategy described above. We started with the minimal hypothesis $n_\omega = 3$. Parameter γ has been set $\gamma = 4$, which approximately corresponds to the edge of the phase transition for $n = 3$ and $L \geq 30$. Even if for smaller values of L our strategy foresees larger values of γ, it has been found that T^4 was able to find a solution, with this

setting, also for many problems where $20 \leq L \leq 30$. Afterwards more expensive hypotheses has been progressively considered firstly increasing γ, until the value foreseen by the phase transition has been reached (or an excessive cost was foreseen) and then setting $n_\omega = 4$. A value $K = 1\%$ of T_α has been used everywhere. The results are reported in Figure 8. Comparing to Figure 3, we see that many problems lying in the blind spot have been solved. In practice almost all the problems above the phase transition and the line $L = 18$ have been solved with a complexity larger as FOIL but still affordable. A cluster of 20 Pentium III (800 Mhz) has been used for the experiments, where every single problem required a time ranging from few minutes to several hours (an elapsed time comparable to FOIL on a sequential machine). When the problem was not solved we progressively increased the complexity of the hypothesis increasing γ.

7 Discussion

We have shown that combining stochastic search with local deterministic search it is possible to learn approximated concept descriptions where no known classical algorithm was successful. Even if the algorithm is used under the stringent assumption that a conjunctive concept description exists, it is not difficult to extend it in order to cope with more general concept descriptions. For instance, disjunctive descriptions can be learned by integrating T^4 with a set covering algorithm as it is made in most relational learner [Quinlan, 1990; Botta and Giordana, 1993].

However, this is not the fundamental result that emerges from the framework we propose. In our opinion, the most important outcome is the method for estimating the complexity of a learning problem: given a specific hypothesis about the structure of the concept, we have a method for predicting the expected cost for testing the hypothesis. Moreover, a criterion for deciding on-line when stop testing the hypothesis is provided.

A second important result is a negative one, and concerns the possibility of learning descriptions with many variables. Even under the simplistic assumption that all predicates are relevant to the concept description, the task looks hard for many concepts requiring at least 4 variables. Increasing the number of variables, the complexity rises up exponentially. Considering the presence of irrelevant predicates, the analysis we performed still holds, but the the density of sub-formulas of the target concept close to the phase transition becomes even more tiny, and so the difficulty will increase further.

References

[Anglano *et al.*, 1998] C. Anglano, A. Giordana, G. Lobello, and L. Saitta. An experimental evaluation of coevolutive concept learning. In *Proceedings of the 15th International Conference on Machine Learning*, pages 19–23, Madison, WI, July 1998.

[Botta and Giordana, 1993] M. Botta and A. Giordana. SMART+: A multi-strategy learning tool. In *IJCAI-93, Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 937–943, Chambéry, France, 1993.

[Botta *et al.*, 1999] M. Botta, A. Giordana, and L. Saitta. An experimental study of phase transitions in matching. In *Proceedings of th 16th International Joint Conference on Artificial Intelligence*, pages 1198–1203, Stockholm, Sweden, 1999.

[Brassard and Bratley, 1988] G. Brassard and P. Bratley. *Algorithmics: Theory and Practice*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[Giordana and Saitta, 2000] A. Giordana and L. Saitta. Phase transitions in relational learning. *Machine Learning*, x:to appear, 2000.

[Giordana *et al.*, 2000] A. Giordana, L. Saitta, and M. Sebag amd M. Botta. An experimental study of phase transitions in matching. In *Proceedings of th 15th International Conference on Machine Learning*, Stanford, CA, 2000.

[Hogg *et al.*, 1996] T. Hogg, B.A. Huberman, and C.P. Williams, editors. *Artificial Intelligence: Special Issue on Frontiers in Problem Solving: Phase Transitions and Complexity*, volume 81(1-2). Elsevier, 1996.

[Mitchell, 1997] T.M. Mitchell. *Mchine Learning*. McGrow Hill, 1997.

[Muggleton and De Raedt, 1994] S. Muggleton and L. De Raedt. Inductive logic programming: theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[Prosser, 1996] P. Prosser. An empirical study of phase transitions in binary constraint satisfaction problems. *Artificial Intelligence*, 81:81–110, 1996.

[Quinlan, 1990] R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[Rissanen, 1978] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[Serra, 2000] A. Serra. Apprendimento di concetti relazionali. Technical Report TR00-12, DISTA, Universita‘ del Pieminte Orietale, Feb 2000.

[Smith and Dyer, 1996] B.M. Smith and M.E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81:155–181, 1996.

[Williams and Hogg, 1994] C.P. Williams and T. Hogg. Exploiting the deep structure of constraint problems. *Artificial Intelligence*, 70:73–117, 1994.

A Simple Additive Re-weighting Strategy for Improving Margins

Fabio Aiolli and **Alessandro Sperduti**
Department of Computer Science, Corso Italia 40, Pisa, Italy
e-mail: {aiolli, perso}@di.unipi.it

Abstract

We present a sample re-weighting scheme inspired by recent results in margin theory. The basic idea is to add to the training set replicas of samples which are not classified with a sufficient margin. We prove the convergence of the input distribution obtained in this way. As study case, we consider an instance of the scheme involving a 1-NN classifier implementing a Vector Quantization algorithm that accommodates tangent distance models. The tangent distance models created in this way have shown a significant improvement in generalization power with respect to the standard tangent models. Moreover, the obtained models were able to outperform state of the art algorithms, such as SVM.

1 Introduction

In this paper we introduce a simple additive re-weighting method that is able to improve the margin distribution on the training set. Recent results in computational learning theory [Vapnik, 1998; Schapire *et al.*, 1998; Bartlett, 1998] have tightly linked the expected risk of a classifier (i.e. the probability of misclassification of a pattern drawn from an independent random distribution), with the distribution of the margins in the training set. In general, it results that we can expect best performances on generalization (minimal error on test data) when most of the patterns have high margins.

The aforementioned results are at the basis of the theory of two of the most impressive algorithms: Support Vector Machines and Boosting. Either SVM's and Boosting effectiveness is largely due to the fact that they, directly or not, effectively improve the margins on the training set. In particular, SVM explicitly finds the hyper-plane with the largest minimum margin in a dimensional-augmented space where training points are mapped by a kernel function. In this case, margin theory permits to explain impressive performances even in very high dimensional spaces where data are supposed to be more separated. Most of the recent efforts in SVMs are in the choice of the right kernels for particular applications. For example, in OCR problems, the polynomial kernel was proven to be very effective.

On the other side, boosting algorithms, and in particular the most famous version AdaBoost, produce weighted ensemble of hypotheses, each one trained in such a way to minimize the empirical error in a given "difficult" distribution of the training set. Again, it has been shown [Schapire, 1999] that boosting essentially is a procedure for finding a linear combination of weak hypotheses which minimizes a particular loss function dependent on the margins on the training set, literally $Loss = \sum_i \exp(-\mu_i)$. Recently, research efforts related to boosting algorithms faced the direct optimization of the margins on the training set. For example, this has been done by defining different margin-based cost functions and searching for combinations of weak hypotheses so to minimize these functions [Mason *et al.*, 1998].

We will follow a related approach that aims to find a single (eventually non linear) optimal hypothesis where the optimality is defined in terms of a loss-function dependent on the distribution of the margins on the training set. In order to minimize this loss we propose a re-weighting algorithm that maintains a set of weights associated with the patterns in the training set. The weight associated to a pattern is iteratively updated when the margin of the current hypothesis does not reach a predefined threshold on it. In this way a new distribution on the training data will be induced. Furthermore, a new hypothesis is then computed that improves the expectation of the margin on the new distribution. In the following we prove that the distribution converges to a uniform distribution on a subset of the training set.

We apply the above scheme to an OCR pattern recognition problem, where the classification is based on a 1-NN tangent distance classifier [Simard *et al.*, 1993], obtaining a significant improvement in generalization. Basically, the algorithm builds a set of models for each class by an extended version of the Learning Vector Quantization procedure (LVQ [Kohonen *et al.*, 1996]) adapted to tangent distance. In the following we will refer to this new algorithm as Tangent Vector Quantization (TVQ).

The paper is organized as follows. In Section 2, we introduce the concept of margin regularization via the input distribution on the training set. Specifically, we present the Θ-Margin Re-weighting Strategy, which holds the property to guarantee the convergence of the input distribution. In Section 3, we introduce a definition for the margins in a 1-NN scheme that considers the discriminative ratio observed for a particular pattern, and in Section 4 we define the TVQ algorithm. Finally, in Section 5 we present empirical results

comparing TVQ with other 1-NN based algorithms, including SVM.

2 Regularization of the margins

When learning takes place, the examples tend to influence in a different way the discriminant function of a classifier. A discriminant function can be viewed as a resource that has to be shared among different clients (the examples). Often, when pure Empirical Risk Minimization (ERM) principle is applied, that resource is used in a wrong way since, with high probability, it is almost entirely used by a fraction of the training set. Margin theory formally tells us that it is preferable to regularize the discriminant function in such a way to make the examples sharing more equally its support.

Inspired on the basic ideas of margin optimization, here, we propose a simple general procedure applicable, eventually, to any ERM-based algorithm. It permits to regularize the parameters of a discriminant function so to obtain hypotheses with large margins for many examples in the training set.

Without generality loss we consider the margin for a training example as a real number, taking values in $[-1,+1]$, representing a measure of the confidence shown by a classifier in the prediction of the correct label. In a binary classifier, e.g. the perceptron, the margin is usually defined as $yf(x)$ where y is the target and $f(x)$ is the output computed by the classifier. Anyway, it can be easily re-conduced to the $[-1,+1]$ range by a monotonic (linear or sigmoidal) transformation of the output. In any case, a positive value of the margin must correspond to a correct classification of the example.

Given the function $\mu_h(x_i)$ that, provided an hypothesis h, associates to each pattern its margin, we want to define a loss-function that, when minimized, permits to obtain hypotheses with large margins (greater than a fixed threshold θ) for many examples in the training set. For this, we propose to minimize a function that, basically, is a re-formulation of SVM's slack variables:

$$L = \sum_{x_i \in S} (\theta - \mu_h(x_i)) 1_{[\theta - \mu_h(x_i)]}, \tag{1}$$

where S is a training set with N examples, and $1_{[\theta-\mu_h(x_i)]} = 1$ if $\mu_h(x_i) < \theta$, and 0 otherwise. The L function is null for margins higher than the threshold θ and is linear with respect to the values of the margins when they are below this threshold.

We suggest to minimize L indirectly via a two-step iterative method that "simultaneously" (1) searches for an a priori distribution $\{\gamma_i\}$ for the examples that, given the current hypothesis h, better approximates the function $1_{[\theta-\mu_h(x)]}$ and (2) searches for a hypothesis h (e.g. by a gradient based procedure) that, provided the distribution $\{\gamma_i\}$, improves the weighted function

$$H = \sum_{p=1}^{N} \gamma_p \mu_h(x_p). \tag{2}$$

This new formulation is equivalent to that given in eq. (1) provided that $\{\gamma_i\}$ converges to the uniform distribution on the θ-mistakes (patterns that have margin less than the threshold).

Θ-Margin Re-weighting Strategy

Input:
T: number of iterations;
$\mathcal{H}$: hypotheses space;
θ: margin threshold;
$k(\cdot) \in (0, K]$: bounded function;
$S = \{(x_i, y_i)\}_{i=1,..,N}$: training set;

Initialize
$h_0 \in \mathcal{H}$ (initial hypothesis);
for $i = 1, .., N$
$w_i(0) \leftarrow 1, \gamma_i(0) \leftarrow \frac{1}{N}$;

for $t = 1, .., T$

begin

find h_t such that
$\sum_{i=1}^{N} \gamma_i(t-1)\mu_{h_t}(x_i) > \sum_{i=1}^{N} \gamma_i(t-1)\mu_{h_{t-1}}(x_i)$;
$w_i(t) \leftarrow w_i(t-1) + k(t) 1_{[\theta - \mu_{h_t}(x_i)]}$;
$\gamma_i(t) \leftarrow \frac{w_i(t)}{\sum_{j=1}^{N} w_j(t)}$;

end

return h_T;

Figure 1: The Θ-Margin Re-weighting Strategy.

The algorithm, shown in Figure 1, consists of a series of trials. An optimization process, that explicitly maximizes the function $\mu_h(x)$ according to the current distribution for the examples, works on an artificial training set S', initialized to be equal to the original training set S. For each t, $k(t)$ replicas of those patterns in S that have margin below the fixed threshold θ are added to S' augmenting their density in S' and consequently their contribution in the optimization process. Note that $w_i(t)$ denotes the number of occurrences in the extended training set S' of the pattern x_i.

In the following, we will prove that the simple iterative procedure just described makes the distribution approaching a uniform distribution on the θ-mistakes, provided that $k(t)$ is bounded.

2.1 Convergence of the distribution

For each trial t, given the margin of each example in the training set S, we can partition the training sample as $S = S_M(t) \cup S_C(t)$ where $S_M(t)$ is the set of θ-mistakes and $S_C(t) = S - S_M(t)$ is the complementary set of θ-correct patterns.

Let denote $W(t) = |S'(t)|$ and let $w_i(t)$ be the number of occurrences of pattern x_i in S' at time t, with density $\gamma_i(t) = w_i(t)/W(t)$. Moreover, let $\Lambda(t)$ be a suitable function of t such that $W(t+1) = W(t)\Lambda(t)$.
Let $w_i(t+1) = w_i(t) + k(t)$ be the update rule for the number of occurences in S', where $k(t)$ is bounded and takes values in $(0, K]$ (note that $k(t)$ may change at different iterations but it is independent from i). It's easy to verify that $\Lambda(t) \geq 1$ for each t because of the monotonicity of $W(t)$, and that $\Lambda(t) \rightarrow$

1 with the number of iterations. In fact

$$1 \leq \frac{W(t+1)}{W(t)} = \frac{W(t)+\Delta W(t)}{W(t)} \leq 1 + \frac{K|S|}{W(t)} \rightarrow 1.$$

At time $t+1$ we have

$$\gamma_i(t+1) = \frac{w_i(t) + k(t)1_{[\theta-\mu(t)]}}{\Lambda(t)W(t)} = \frac{\gamma_i(t) + \frac{k(t)}{W(t)}1_{[\theta-\mu(t)]}}{\Lambda(t)}.$$

First of all we show that the distribution converges. This can be shown by demonstrating that the changes tend to zero with the number of iterations, i.e., $\forall x_i \in S, |\Delta\gamma_i(t)| \rightarrow 0$.

We have

$$\Delta\gamma_i(t) = \frac{\frac{k(t)}{W(t)}1_{[\theta-\mu(t)]} - (\Lambda(t)-1)\gamma_i(t)}{\Lambda(t)},$$

which can be easily bounded in module by a quantity that tends to 0:

$$|\Delta\gamma_i(t)| \leq \frac{\frac{K}{W(t)} + (\Lambda(t)-1)\gamma_i(t)}{\Lambda(t)} \rightarrow 0.$$

We now show to which values they converge. Let m_i and ϵ_i be, respectively, the cumulative number and the mean ratio of θ-mistakes for x_i on the first t epochs, $\epsilon_i = \frac{m_i(t)}{t}$, then

$$\begin{aligned}\gamma_i(t) &= \frac{w_i(0) + E[k(t)]m_i(t)}{W(0) + E[k(t)]\sum_{j=1}^{N} m_j(t)} \\ &= \frac{1 + tE[k(t)]\epsilon_i}{N + tE[k(t)]\sum_{j=1}^{N}\epsilon_j} \rightarrow \frac{\epsilon_i}{\sum_{j=1}^{N}\epsilon_j}.\end{aligned}$$

Given the convergence of the optimization process that maximizes H in eq. (2), the two sets S_M and S_C are going to become stable and the distribution on S will tend to a uniform distribution in S_M (where $\epsilon_i \rightarrow 1$) and will be null elsewhere (where $\epsilon_i \rightarrow 0$). This can be understood in the following way as well.

Given the definition of the changes made on the gamma values on each iteration of the algorithm, we calculate the function that we indeed minimize. Since $\Delta w(t) = k(t) \cdot 1_{[\theta-\mu(t)]}$, after some algebra, we can rewrite $\Delta\gamma_i(t)$ as:

$$\Delta\gamma_i(t) = \frac{\Delta W(t)}{W(t+1)}\left(\frac{1_{[\theta-\mu_i(t)]}}{|S_M(t)|} - \gamma_i(t)\right).$$

Thus, $\Delta\gamma_i(t) = 0$ when $\gamma_i = \frac{1}{|S_M(t)|}1_{[\theta-\mu_i(t)]}$, for which the minimum of function

$$E_\gamma = \frac{\Delta W(t)}{W(t+1)}\left(\frac{1}{2}\sum_i \gamma_i^2(t) - \frac{1}{|S_M(t)|}\sum_i 1_{[\theta-\mu_i(t)]}\gamma_i(t)\right)$$

is reached. Note that, the minimum of E_γ is consistent with the constraint $\sum_i \gamma_i = 1$.

In general, the energy function is modulated by a term decreasing with the number of iterations, dependent on the $k(t)$ used but independent from gamma, that can be viewed as a sort of annealing introduced in the process.

In the following, we study a specific instance of the Θ-Margin Re-weighting Strategy.

3 Margins in a 1-NN framework and tangent distance

Given a training example $(x_i, y_i) \in S$, $y_i \in Y$ and a fixed number of models for each class, below, we give a definition of the margin for the example when classified by a distance based 1-NN classifier.

Given the example (x_i, y_i), let z_i^p and z_i^n be the squared distances between the nearest of the positive set of models and the nearest of the negative sets of models, respectively. We can define the margin of a pattern in the training set as:

$$\mu_i = \frac{z_i^n - z_i^p}{z_i^n + z_i^p}. \tag{3}$$

This formula takes values in the interval $[-1, +1]$ representing the confidence in the prediction of the 1-NN classifier. Higher values of the μ_i's can also be viewed as an indication of a higher discriminative power of the set of models with respect to the pattern. Moreover, a pattern will result correctly classified in the 1-NN scheme if and only if its margin is greater than zero.

In this paper, we are particularly interested in distances that are invariant to given transformations. Specifically, we refer to the *one-sided tangent distance* [Simard, 1994; Hastie *et al.*, 1995; Schwenk and Milgram, 1995b; 1995a], which computes the distance between a pattern x_1 and a pattern x_2 as the minimum distance between x_1 and the linear subspace $\tilde{x}_2(\alpha)$ approximating the manifold induced by transforming the pattern x_2 according to a given set of transformations:

$$d_T(x_1, x_2) = \min_\alpha \|x_1 - \tilde{x}_2(\alpha)\|. \tag{4}$$

If the transformations are not known a priori, we can learn them by defining, for each class y, a (one-sided) tangent distance model M_y, compounded by a *centroid* C_y (i.e., a prototype vector for class y) and a set of *tangent vectors* $\mathcal{T}_y = \{T_k\}$ (i.e., an orthonormal base of the linear subspace $\tilde{C}_y(\alpha)$), that can be written as $M_y(\alpha) = C_y + \sum_{k=1}^{|\mathcal{T}_y|}\alpha_k T_k$.

This model can be determined by just using the N_y positive examples of class y, i.e. $(x_i, y) \in S$, as

$$M_y = \arg\min_M \sum_{p=1}^{N_y} \min_{\theta_p} \|M(\theta_p) - x_p\|^2, \tag{5}$$

which can easily be solved by resorting to principal component analysis (PCA) theory, also called *Karhunen-Loéve Expansion*. In fact, equation (5) can be minimized by choosing C_y as the average over all available positive samples x_p, and $\mathcal{T}_y$ as the set of the most representative eigenvectors (principal components) of the covariance matrix $\Sigma_y = \frac{1}{N_y}\sum_{p=1}^{N_y}(x_p - C_y)(x_p - C_y)^t$.

The corresponding problem for the two-sided tangent distance can be solved by an iterative algorithm, called (two-sided) HSS, based on Singular Value Decomposition, proposed by Hastie et al. [Hastie *et al.*, 1995]. When the one-sided version of tangent distance is used, HSS and PCA coincide. So, in the following, the one sided version of this algorithm will be simply referred as to HSS.

Given a one-sided tangent distance model M_y, it is quite easy to verify that the squared tangent distance between a pattern x and the model M_y can be written as:

$$d_T^2(x, M_y) = \delta^t \delta - \sum_{k=1}^{|\mathcal{T}|} \alpha_k^2 \quad (6)$$

where $\delta = x - C_y$, $\alpha_k = \delta^t T_k$ and δ^t denotes the transpose of δ. Consequently, in our definition of margin, we have $z_i^p = \min_{M_y \in y_i} d_T^2(x_i, M_y)$, and $z_i^n = \min_{M_y \notin y_i} d_T^2(x_i, M_y)$.

Given this definition of margin, we can implement the choice of the new hypothesis in the Θ-Margin Re-weighting Strategy by maximizing the margin using gradient ascent on the current input distribution.

3.1 Improving margins as a driven gradient ascent

Considering the tangent distance formulation as given in equation (6) we can verify that it is defined by scalar products. Thus, we can derivate it with respect to the centroid C and the tangent vectors $\mathcal{T} = \{T_k\}$ of the nearest positive model obtaining:

$$\frac{\delta z_i^p}{\delta C} = -2(\delta - \sum_{k=1}^{|\mathcal{T}|} \alpha_k T_k), \qquad \frac{\delta z_i^p}{\delta T_k} = -2\alpha_k \delta.$$

Considering that $\frac{\delta \mu_i}{\delta C} = \frac{\delta \mu_i}{\delta z_i^p}\frac{\delta z_i^p}{\delta C}$ and $\frac{\delta \mu_i}{\delta T_k} = \frac{\delta \mu_i}{\delta z_i^p}\frac{\delta z_i^p}{\delta T_k}$ we can compute the derivative of the margin with respect to changes in the nearest positive model:

$$\begin{aligned} \frac{\delta \mu_i}{\delta C} &= 4\frac{z_i^n}{(z_i^n + z_i^p)^2}(\delta - \sum_{k=1}^{|\mathcal{T}|} \alpha_k T_k), \\ \frac{\delta \mu_i}{\delta T_k} &= 4\frac{z_i^n}{(z_i^n + z_i^p)^2}\alpha_k \delta. \end{aligned}$$

A similar solution is obtained for the nearest negative model since it only differs in changing the sign and in exchanging indexes n and p. Moreover, the derivatives are null for all the other models.

Thus, we can easily maximize the average margin in the training set if for each pattern presented to the classifier we move the nearest models in the direction suggested by the gradient. Note that, like in the LVQ algorithm, for each training example, only the nearest models are changed.

When maximizing the expected margin on the current distribution $\{\gamma_i\}$, i.e., H, for each model $M = (C, \{T_k\})$ we have:

$$\Delta C = \eta \sum_{i=1}^{|S|} \gamma_i \frac{\delta \mu_i}{\delta C}, \qquad \Delta T_k = \eta \sum_{i=1}^{|S|} \gamma_i \frac{\delta \mu_i}{\delta T_k},$$

where η is the usual learning rate parameter. In the algorithm (see Figure 2), for brevity, we will group the above variations by referring to the whole model, i.e., $\Delta M = \eta \sum_{i=1}^{|S|} \gamma_i \frac{\delta \mu_i}{\delta M}$.

TVQ Algorithm

Input:
T: no. of iterations;
Q: no. of models per class;
θ: margin threshold;

Initialize
$W \leftarrow N$, $\gamma_i \leftarrow \frac{1}{N}$;
$\forall y, q$, initialize $M_y^{(q)} \leftarrow (C_y^{(q)}, \mathcal{T}_y^{(q)})$ with random models;

for $t = 1, .., T$
$\forall y, \forall q$, $\Delta M_y^{(q)} = 0$;
$\forall (x_i, y_i) \in S$, select $(q_p, \bar{y}_i, q_n)$ s.t. $M_{y_i}^{(q_p)}$ and $M_{\bar{y}_i}^{(q_n)}$ are the nearest, positive and negative, models. Compute μ_i as in eq. (3) and accumulate the changes on the nearest models

$$\Delta M_{y_i}^{(q_p)} \leftarrow \Delta M_{y_i}^{(q_p)} + \gamma_i \frac{\delta \mu_i}{\delta M_{y_i}^{(q_p)}};$$

$$\Delta M_{\bar{y}_i}^{(q_n)} \leftarrow \Delta M_{\bar{y}_i}^{(q_n)} + \gamma_i \frac{\delta \mu_i}{\delta M_{\bar{y}_i}^{(q_n)}};$$

$\forall y, \forall q$, $M_y^{(q)} = M_y^{(q)} + \eta \Delta M_y^{(q)}$ and orthonormalize its tangents;
$\forall (x_i, y_i) \in S$, update the distribution γ_i by the rule

$$\gamma_i \leftarrow \gamma_i + \frac{1_{[\theta - \mu_i]}}{W}$$

Normalize γ_i's such that $\sum_{i=1}^{N} \gamma_i = 1$;
$W \leftarrow W + |\{(x_i, y_i) | \mu_i < \theta\}|$;
End

Figure 2: The TVQ Algorithm.

4 The TVQ algorithm

The algorithm (see Figure 2) starts with random models and a uniform distribution on the training set. For each pattern, the variation on the closest positive and the closest negative models are computed accordingly to the density of that pattern on the training set S'. When all the patterns in S are processed, the models are updated performing a weighted gradient ascent on the values of the margin. Moreover, for each pattern in the training set such that the value of the margin is smaller than a fixed value, the distribution is augmented. The effect is to force the gradient ascent to concentrate on hardest examples in the training set. As we saw in Section 2 the increment to the distribution is simply the effect of adding a replica ($k(t) = 1$) of incorrectly classified patterns to the augmented training set.

The initialization of the algorithm may be done in different ways. The default choice is to use random generated models however, when the training set size is not prohibitive, we can drastically speed up the algorithm by taking as initial models the ones generated by any algorithm (e.g., HSS). How-

Method	Parameters	Err%
HSS 1-sided	15 tangents	3.58
LVQ 2.1	16 codebooks	3.52
TD-Neuron	15 tangents	3.51
HSS 2-sided	9 tangents	3.40
Euclidean 1-NN	prototypes	3.16
SVM	Linear	10.64
SVM	Poly d=2	2.82
SVM	Poly d=3	3.23
SVM	Poly d=4	4.02

Table 1: Test results for different 1-NN methods.

ever, in the case of multiple models per class the initialization through the HSS method would generate identical models for each class and that would invalidate the procedure. A possible choice in this case, is to generate HSS models by using different random conditional distributions for different models associated to the same class. Another solution, which is useful when the size of the training set is relatively large, is to initialize the centroids as the average of the positive instances and then generating random tangents. Experimental results have shown that the differences on the performance obtained by using different initialization criteria are negligible. As we could expect the speed of convergence with different initialization methods may be drastically different. This is due to the fact that when TVQ is initialized with HSS models it starts with a good approximation of the optimal hypothesis (see Figure 3-(a)), while random initializations implicitly introduce an initial poor estimate of the final distribution due to the mistakes that most of the examples do on the first few iterations.

5 Results

We compared the TVQ algorithm versus SVMs and other 1-NN based algorithms: 1-sided HSS, 2-sided HSS, TD-Neuron [Sona *et al.*, 2000], and LVQ. The comparison was performed using exactly the same split of a dataset consisting of 10705 digits randomly taken from the NIST-3 dataset. The binary 128x128 digits were transformed into 64-grey level 16x16 images by a simple local counting procedure[1]. The only preprocessing performed was the elimination of empty borders. The training set consisted of 5000 randomly chosen digits, while the remaining digits were used in the test set.

The obtained results for the test data are summarized in Table 1. For each algorithm, we reported the best result, without rejection, obtained for the dataset. Specifically, for the SVM training we used the SVM^{Light} package available on the internet[2]. Different kernels were considered for the SVMs: linear and polynomial with degrees 2,3 and 4 (we used the default for the other parameters). Since SVMs are binary classifiers, we built 10 SVMs, one for each class against all the others, and we considered the overall prediction as the label with higher margin. The best performance has been obtained

[1]The number of pixel with value equal to 1 is used as the grey value for the corresponding pixel in the new image.

[2]http://www-ai.cs.uni-dortmund.de/SOFTWARE/SVM_LIGHT/

	$\theta = 0.3$		$\theta = 0.4$		$\theta = 0.1$
Q	$\lvert\mathcal{T}\rvert$ =10	$\lvert\mathcal{T}\rvert$ = 15	$\lvert\mathcal{T}\rvert$ =10	$\lvert\mathcal{T}\rvert$ =15	$\lvert\mathcal{T}\rvert$ = 0
1	2.40	2.20	3.00	2.22	6.40
3	2.10	2.26	2.43	2.10	–

Table 2: Test results for TVQ.

with a polynomial kernel of degree 2. We ran the TVQ algorithm with two different values for θ and four different architectures. Moreover, we ran also an experiment just using a single centroid for class (i.e., $|\mathcal{T}_y| = 0$) with $\theta = 0.1$. The smaller value for θ has been chosen just to account for the far smaller complexity of the model.

In almost all the experiments the TVQ algorithm obtained the best performance. Results on the test data are reported in Table 2. Specifically, the best result for SVM is worst than almost all the results obtained with TVQ. Particularly interesting is the result obtained by just using a single centroid for each class. This corresponds to perform an LVQ with just 10 codebooks, one for each class.

In addition, TVQ returns far more compact models allowing a reduced response time in classification. In fact, the 1-NN using polynomial SVMs with $d = 2$, needs 2523 support vectors, while in the worst case the models returned by the TVQ involve a total of 480 vectors (one centroid plus 15 tangents for each model).

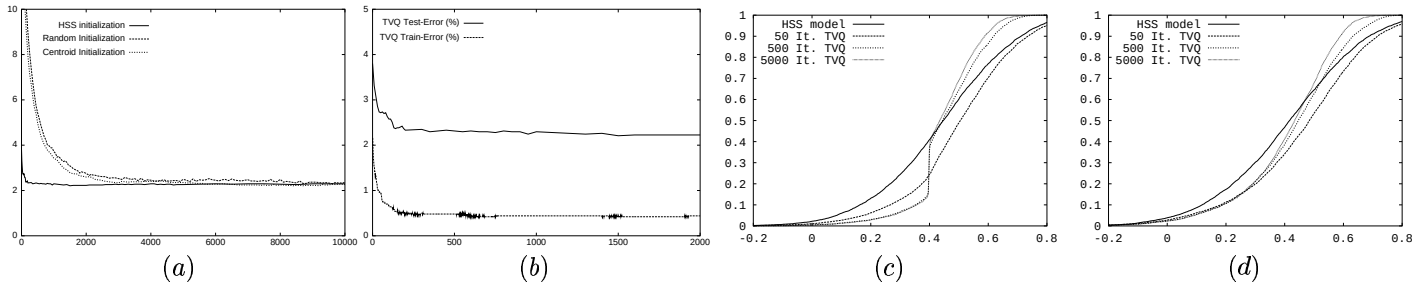
In Figure 3, typical error curves for the training and test errors (3-(b)), as well as the margin distributions on the training set (3-(c)) and the induced margin distribution on the test set (3-(d)) are reported. From these plots it is easy to see that the TVQ doesn't show overfitting. This was also confirmed by the experiments involving the models with higher complexity and smaller values of θ. Moreover, the impact of the θ-margin on the final margin distribution on the training set is clearly shown in 3-(c), where a steep increase of the distribution is observed in correspondence of θ at the expenses of higher values of margin. Even if at a minor extent, a similar impact on the margin distribution is observed for the test data.

In Figure 4 we have reported the rejection curves for the different algorithms. As expected, the TVQ algorithm was competitive with the best SVM, resulting to be the best algorithm for almost the whole error range.

6 Conclusions

We proposed a provably convergent re-weighting scheme for improving margins, which focuses on "difficult" examples. On the basis of this general approach, we defined a Vector Quantization algorithm based on tangent distance, which experimentally outperformed state of the art classifiers both in generalization and model compactness. These results confirm that the control of the shape of the margin distribution has a great effect on the generalization performance.

When comparing the proposed approach with SVM, we may observe that, while our approach shares with SVM the Statistical Learning Theory concept of uniform convergence of the empirical risk to the ideal risk, it exploits the input distribution to directly work on non-linear models instead of resorting to predefined kernels. This way to proceed is



(a) (b) (c) (d)

Figure 3: TVQ with 15×1 tangents, and $\theta = 0.4$: (a) comparison with different initialization methods; (b) test and training error; (c) cumulative margins on the training set at different iterations; (d) cumulative margins on the test set at different iterations.

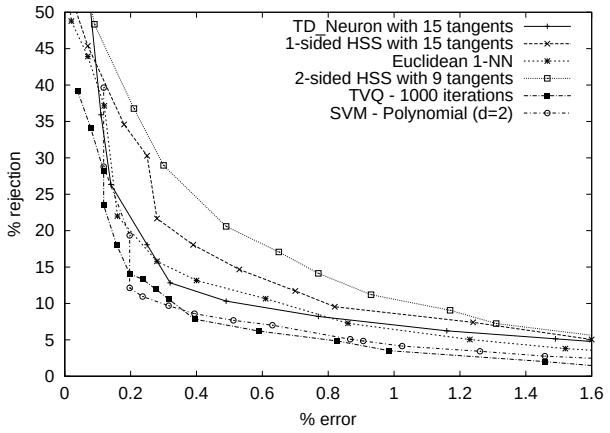


Figure 4: Detail of rejection curves for the different 1-NN methods. The rejection criterion is the difference between the distances of the input pattern with respect to the first and the second nearest models.

very similar to the approach adopted by Boosting algorithms. However, in Boosting algorithms, several hypotheses are generated and combined, while in our approach the focus is on a single hypothesis. This justifies the adoption of an additive re-weighting scheme, instead of a multiplicative scheme which is more appropriate for committee machines.

Acknowledgments

Fabio Aiolli wishes to thank Centro META - Consorzio Pisa Ricerche for supporting his PhD fellowship.

References

[Bartlett, 1998] P.L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Trans. on Infor. Theory*, 44(2):525–536, 1998.

[Hastie *et al.*, 1995] T. Hastie, P. Y. Simard, and E. Säckinger. Learning prototype models for tangent distance. In G. Teasauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neur. Inform. Proc. Systems*, volume 7, pages 999–1006. MIT Press, 1995.

[Kohonen *et al.*, 1996] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola. Lvq_pak: The learning vector quantization program package. Technical Report A30, Helsinki Univ. of Tech., Lab. of Computer and Inform. Sci., January 1996.

[Mason *et al.*, 1998] L. Mason, P. Bartlett, and J. Baxter. Improved generalization through explicit optimization of margins. Technical report, Dept. of Sys. Eng., Australian National University, 1998.

[Schapire *et al.*, 1998] R.E Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *An. of Stat.*, 26(5), 1998.

[Schapire, 1999] R. Schapire. Theoretical views of boosting. In *Computational Learning Theory: Proc. of the 4th European Conference, EuroCOLT'99*, 1999.

[Schwenk and Milgram, 1995a] H. Schwenk and M. Milgram. Learning discriminant tangent models for handwritten character recognition. In *Intern. Conf. on Artif. Neur. Netw.*, pages 985–988. Springer-Verlag, 1995.

[Schwenk and Milgram, 1995b] H. Schwenk and M. Milgram. Transformation invariant autoassociation with application to handwritten character recognition. In G. Teasauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neur. Inform. Proc. Systems*, volume 7, pages 991–998. MIT Press, 1995.

[Simard *et al.*, 1993] P. Y. Simard, Y. LeCun, and J. Denker. Efficient pattern recognition using a new transformation distance. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 50–58. Morgan Kaufmann, 1993.

[Simard, 1994] P. Y. Simard. Efficient computation of complex distance metrics using hierarchical filtering. In J. D. Cowan, G. Teasauro, and J. Alspector, editors, *Advances in Neur. Inform. Proc. Systems*, volume 6, pages 168–175. Morgan Kaufmann, 1994.

[Sona *et al.*, 2000] D. Sona, A. Sperduti, and A. Starita. Discriminant pattern recognition using transformation invariant neurons. *Neur. Comput.*, 12(6):1355–1370, 2000.

[Vapnik, 1998] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

MACHINE LEARNING AND DATA MINING

KNOWLEDGE ACQUISITION

Knowledge Analysis on Process Models

Jihie Kim and Yolanda Gil
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292, U.S.A.
jihie@isi.edu, gil@isi.edu

Abstract

Helping end users build and check process models is a challenge for many science and engineering fields. Many AI researchers have investigated useful ways of verifying and validating knowledge bases for ontologies and rules, but it is not easy to directly apply them to checking process models. Other techniques developed for checking and refining planning knowledge tend to focus on automated plan generation rather than helping users author process information. In this paper, we propose a complementary approach which helps users author and check process models. Our system, called KANAL, relates pieces of information in process models among themselves and to the existing KB, analyzing how different pieces of input are put together to achieve some effect. It builds interdependency models from this analysis and uses them to find errors and propose fixes. Our initial evaluation shows that KANAL was able to find most of the errors in the process models and suggest useful fixes including the fixes that directly point to the sources of the errors.

1 Introduction

Building process models is essential in many science and engineering fields. Since some of these processes are quite complex, it is useful to provide tools that enable users to specify process models. Figure 1 shows an example of a process model in Cell Biology to describe how a Lambda virus invades a cell. To be able to specify such a model, the user has to specify each of the individual steps and connect them appropriately. There are different types of connections among the steps, including decomposition links between steps and substeps, ordering constraints, disjunctive alternatives, etc. Even in process models of small size, the number of steps and connections between them is large enough that users would benefit from the assistance of intelligent acquisition tools that help them specify process models correctly. For example, the user may forget to specify the links between the steps, or may specify wrong links. We found such errors in a Biological Weapon production model built by a subject matter expert. Even though it can be considered as a relatively simple

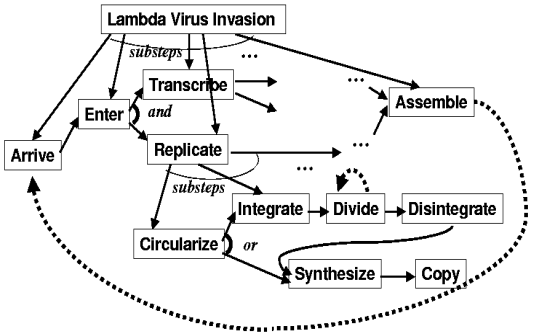


Figure 1: Example Process Model.

model (consisting of 54 steps) and many people had looked at it, there were at least two errors: (a) there were two steps that were both specified as the second substep of the same step; (b) there were two sequential substeps whose ordering information was missing. Although these may be simple errors, in more complex models users may generate more serious problems and have difficulty noticing and fixing them.

Graphical tools to lay out a process model and draw connections among steps abound, but the tools are limited to simple checks on the process models because there is no semantics associated to the individual steps. In contrast, we assume a knowledge-rich environment that enables users to specify what the process and its steps mean, what they should accomplish and how, and what the features of the objects involved in those steps are. In order to check a process model such as the one in the figure thoroughly, the system would have some background knowledge about what a cell is, that it can be entered because it is a physical object with a barrier, etc. It would also be helpful for the tool to know what a change of location is, and that the Enter step implies a change of location of its agent. Domain ontologies and upper or middle level ontologies are commonly used to represent this kind of background knowledge. With this context, the tool can be much more helpful in checking that the process model makes sense within the background knowledge that it has.

Past research in validation and verification of knowledge bases addresses the detection of errors in rule bases [Preece & Shinghal, 1994; O'Keefe & O'Leary, 1994] or in ontologies [McGuinness *et al.*, 2000] and has not addressed pro-

cess models specifically. Research on interactive tools to acquire planning knowledge is also related [Chien, 1998; Myers, 1996; Huffman & Laird, 1995], but their focus is on acquiring knowledge about how to generate plans instead of acquiring the specific plans themselves. Plan authoring tools are closer in spirit to the process authoring tools that we are aiming for, and as such KANAL could be used to check errors in plans produced by plan editing tools. Formal analyses of partial-order and hierarchical planning algorithms define some desirable properties of plans, such as justifiability and correctness [Kambhampati, Knoblock, & Yang, 1995; Yang, 1990; Tate, 1996]. Generative planners such as SIPE, NOAH, and NONLIN [Wilkins, 1988; Sacerdoti, 1977; Tate, 1977] use critics that detect problems in the plans that they generate while planning. Much of the work on planning does not exploit background knowledge and ontologies, which we believe is crucial technology to advance the state of the art in process modeling.

We have developed a tool that checks process models specified by a user, reports possible errors in the models, and generates specific suggestions to the user about how to fix those errors. Our system is called KANAL (Knowledge ANALysis), and it helps users build or modify process models by detecting invalid statements and pointing out what additional knowledge needs to be acquired or what existing knowledge needs to be modified. Our approach is inspired on previous work on EXPECT using *Interdependency Models* [Swartout & Gil, 1995; Kim & Gil, 2000]. These models of the interdependencies between different pieces of knowledge can be derived by analyzing how knowledge is used during problem solving. By analyzing these interdependencies, a knowledge acquisition tool can detect inconsistencies and missing knowledge and alert the user of potential problems in the knowledge base. Finally, based on the context provided by the Interdependency Models the tool can guide the user in fixing these problems by correcting inconsistencies and by adding further knowledge. KANAL analyzes the interdependencies among the individual steps of a process model. For example, a simulation of the execution of the steps allows KANAL to analyze interdependencies between the conditions and effects of different steps, such as that the required conditions for each step are met when the step is supposed to take place, and that the expected effects of the overall process are in fact obtained.

The paper begins by describing the representation of process models assumed in KANAL. Then we describe how Interdependency Models can be applied to check various features of process models. Next, we present the current implementation of KANAL and the algorithms that it uses to detect different kinds of errors. Finally, we present the results from a preliminary evaluation that show that KANAL can detect most of the errors that were randomly introduced in originally error-free process models, suggesting useful fixes that often point directly to the source of the errors.

2 Representing Process Models

This section describes briefly our representation of process models, which is consistent with current efforts on standard languages and process ontologies, such as PDDL [Ghallab *et al.*, 1998] and NIST's PSL [Tissot & Gruninger, 1999].

A process model is composed of a number of (sub)steps. Each individual step has preconditions and effects, where the preconditions specify the conditions needed to be satisfied to activate the step and the effects describe changes that result from the execution of the step. For example, an "Enter" step has a precondition that the objects to enter should be near the entrance of a container object. Its effect can include a location change from outside of a space to inside of the space and also a status change to being contained within the container. These can be represented as a precondition list and add/delete lists (as in STRIPS operators).

The steps within a process model are connected to other steps through different kinds of *links* including:

- decomposition links: Users can specify super-step/substep relations. For example, an Invade step can have Arrive, Enter and Take Control as its substeps, and each of these substeps can have their own substeps.
- temporal links: Users can specify ordering constraints among the steps. For example, in modeling virus invasion, the Take Control step should follow the Enter step.
- disjunctive links: There might be more than one way of performing a given task, and the alternatives can be represented by disjunctive links. For example, the DNA of a Lambda virus can either start its replication right after entering a cell or be integrated with the host chromosome before the replication.
- causal links: If the editor allows users to specify enablement/disablement between steps, since KANAL can compute the actual causal relationships among the steps from the simulation results (by examining the outcome of the steps and the preconditions checked by other steps), the user-specified causal links can be used for validating the model.

Each step can have several *roles*. For example, in an Enter step an object can play the role of an agent and another object can play the role of the container being entered. A general description of an Enter step can be instantiated for the Virus invasion process by *assigning* the concept virus to the agent role of Enter and the concept cell to the container role. These role assignments cause further interdependencies in the knowledge base, since the objects assigned to the roles have their own constraints and definitions that must be consistent with those of the process models and their steps.

3 Acquiring Process Models: The End-to-End System

Currently, KANAL is being developed as a module within an ambitious end-to-end system that will support subject matter experts entering domain knowledge as part of the DARPA Rapid Knowledge Formation (RKF) program. In this project, users will build process models is by using "concept composition" [Clark & Porter, 1997]. Users can build process models by retrieving components (actions and objects) and then connecting them using various kinds of links. The user interface

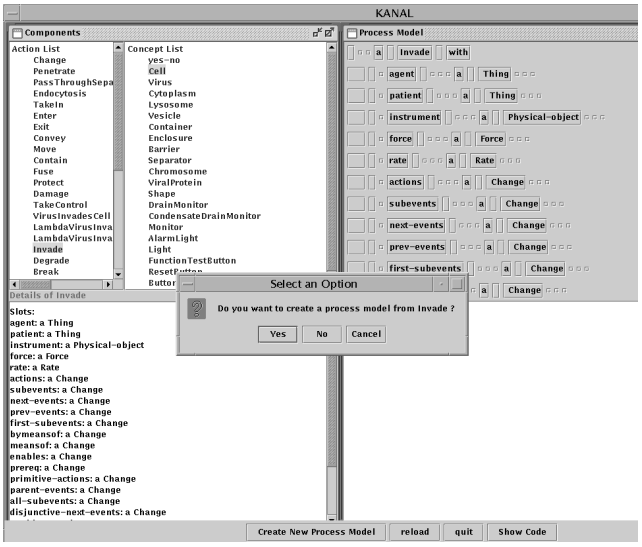


Figure 2: The KANAL interface for building process models.

and the component library have not been fully implemented and integrated with KANAL, although a preliminary version of the three was done to illustrate the approach with a small scale scenario of a process model for virus invasion.

Figure 2 is an interface that we built to show how the component approach can be used to build process models by linking various objects in the knowledge base. The user has selected the Invade component to start building the Virus Invasion model. The "agent" role can be assigned to the Virus concept, the Enter component can be linked as a subevent, and so on.

Although KANAL is built to check process models constructed by concept composition, it can also be used for checking process models in other environments. For example, providing procedural knowledge required by intelligent tutoring systems has been an ongoing challenge, and they sometimes use simulators to build and refine their models [Scholer *et al.*, 2000]. We are also investigating the use of KANAL to help teachers formalize process models that will be used as lessons by a tutoring system.

4 Using Interdependency Models

In past work, Interdependency Models have been successfully used in building and checking problem-solving knowledge in EXPECT [Kim & Gil, 1999; Kim & Gil, 2000]. They have been used in analyzing how individual components of a knowledge base are related and interact when they are used during problem solving. An example of interdependency between two pieces of procedural knowledge is that one may be used by the other to achieve a subgoal. Other kinds of Interdependency Models include interdependencies between factual knowledge and procedural knowledge. Interdependency Models can point out missing pieces in solving a problem and be used to predict what pieces are related and how. In this paper, we show a novel use of Interdependency Models to check process models.

To guide users in developing process models, KANAL builds interdependencies among objects in the knowledge base, and uses them to perform two kinds of checks: static checks and dynamic checks. Static checks are performed by posing questions about various features of the process model, and dynamic checks are performed by simulating the execution of the process model. Our initial work to date has focused on dynamic checks.

In order to perform static checks, we plan to maintain a list of sample query templates, such as retrieving the values of different links, types of roles assigned to steps, etc. A list of instantiated queries can be generated for a particular model using these templates. Users could select key queries from this list and also specify the answers expected from the queries. A trace of the answer to a query can be considered as a model of the interdependencies in that it reflects how different pieces of knowledge are put together to generate the answer.

Dynamic checks can be done on the simulated execution of the process model. The simulation results show how different steps are related to each other, including temporal ordering and causal relationships. The results also show how certain effects are produced by a set of sequences of steps. The resulting Interdependency Model enables checking if all the steps are properly linked, all the preconditions of each step are satisfied during the simulation, all the expected effects can be achieved, there are no unexpected effects, there are no impossible paths, etc. Also the interdependencies can point to potential ways of solving errors and gaps in the model, such as changing ordering constraints to reinstate disabled effects, finding steps that can generate unachieved effects, adding missing links, etc.

Our current work focuses on dynamic checks, using the simulation as a tool to generate Interdependency Models. The next section describes how we check the process models using simulation results.

5 Checking Process Models with KANAL

Our current implementation is built using the KM knowledge representation and reasoning system [Clark & Porter, 2000], and invokes its simulator to generate alternative simulations of a process model. KM's simulation of process models can be seen as a symbolic execution of a linearization of the process model using Skolem instances. KM provides a function that can execute a step in a given situation and create a new situation based on the add/delete lists of the step. KANAL uses this function to execute the steps in the given model and check various kinds of problems. Whenever a precondition test fails or any of the events are *undoable* (a step is undoable when not all of its previous steps were executed), KANAL interrupts the simulation and reports the *unreached* steps (a step is unreached when the simulation stops before the step is simulated). It also reports other problems found until it finishes the simulation. From the simulation results, including the problems detected during the simulation, KANAL can compute potential fixes based on Interdependency Models.

The subsections below describe each type of check that KANAL performs in detail.

5.1 Checking Unachieved Preconditions

A precondition is not achieved either because there is no previous step that produces the needed effect or because some previous steps undo the precondition. For example, an Integrate step of a virus DNA into a host chromosome may undo a precondition (that the viral DNA is exposed) of a step to Synthesize protein from DNA. To be able to synthesize the viral protein needed for the replication, an additional Dis-Integrate step that can reinstate the exposure is required.

The general algorithm to check preconditions is as follows:

1. Detect problem
   (a) Run simulation with Skolem instances
   (b) Collect failed step(s)
   (c) Collect unachieved preconditions of failed step
   (d) Show them to user
2. Help user fix problem
   (a) *Suggest that there are missing steps*:
   - Find components in the knowledge base that have the effects needed as preconditions by the failed step and suggest inserting one of these components somewhere within the current process model before the failed step
   (b) *Suggest that there are missing ordering constraints*:
   - Find steps that were executed before the failed step that may have effects that undid the unachieved preconditions
   - Find steps that follow the failed step and have effects that assert the unachieved precondition and suggest inserting an ordering constraint between those steps and the failed step
   (c) *Suggest modifying the step* whose preconditions were not achieved

For the above type of failure, KANAL suggests (a) adding a Dis-Integrate step, (b) changing or adding ordering constraints for the Integrate step, and (c) deleting or modifying the Synthesize step. Suggestion (a) would be the one that user is looking for in order to fix the problem in this example.

5.2 Checking Expected Effects

KANAL informs the user of the effects of each step during simulation. This allows users to check that the process occurs as they anticipated. In addition, users can specify to KANAL what they expect to be the case after the overall process happens. These *expected effects* are what the user indicates should result from the simulation and/or the post-conditions of the composed process model. For example, the user may expect that after a virus invasion of a cell, the viral nucleic acid should be located inside the cell. These expected effects can be checked by looking at the results from the simulation. The simulation results are represented as an accumulation of added and deleted facts. Since there may be multiple disjunctive branches in the model, the results from different paths are accumulated separately. KANAL checks the results from each path to see if all of them satisfy the expectation. If there are unachieved effects, KANAL can propose either to add new steps that would achieve them or to modify existing steps.

Currently KANAL checks for expected effects, but does not check explicitly for unexpected effects. We are planning to highlight any unexpected effects to let the users examine whether they should in fact occur.

The algorithm is as follows:

1. Ask user to specify expected effects
2. Detect problem
   (a) Run simulation with Skolem instances
   (b) Collect unachieved effects from each path and record the steps in the failed paths
   (c) Show them to user
3. Help user fix problem
   (a) *Suggest that there are missing steps:*
   - Find components in the knowledge base that have the effects needed and suggest inserting one of these components somewhere within the current process model
   (b) *Suggest modifying steps*:
   - Find steps that may have effects that can potentially change the role values of the unachieved effects and suggest modifying those steps to achieve the effects needed
   (c) *Suggest that there are missing ordering constraints*:
   - Find steps that may have effects that undid the expected effects and find actions that assert the expected effects and suggest inserting an ordering constraint in order to maintain the expected effect where needed

Following the example above, suppose that the user specifies that after the invasion occurs the viral nucleic acid should be located inside the cell. Suppose also that the user forgot to add the Enter step in the model of virus invasion. KANAL suggests (a) adding new steps, such as Move or Enter, that would change the location of the virus, (b) modifying Arrive since it is an existing step that causes the virus to change location, or (c) changing/adding ordering constraints among these steps. The user would choose option (a) to add an Enter step, and the problem would be fixed.

5.3 Checking Unordered Steps

Sometimes the user may either forget to specify links between the steps, or may specify wrong links as in the Biological Weapon production example mentioned in the introduction. These problems may be detected by mapping the steps to the components in the knowledge base that have certain ordering constraints already specified for their substeps, or by running simulation and doing the checks as describe above for unachieved preconditions and effects. During the simulation, KANAL walks through the steps and substeps using the user specified decomposition links and ordering constraints. The simulation is interrupted if some steps cannot be reached because of lack of ordering constraints or the steps are undoable. KANAL highlights these problems and proposes changing or adding ordering constraints among the steps. (We do not show the detailed algorithm here because of lack of space.)

5.4 Checking Inappropriate Execution of Steps

KANAL can also find modeling errors by watching the execution of steps. For example, if some of the assertions to be deleted by a step are not true in the situation where the step is executed, these assertions are reported. Also, if a step produces no effect, i.e., it does not delete or add any assertions, then KANAL reports such problem as well. This type of problem can occur when the step's roles or attributes are assigned to the wrong objects during the composition. Also, if its previous steps have incorrect assignments already and produced unexpected effects, then the following steps cannot be executed appropriately.

KANAL proposes modifying the steps by changing their role assignments or modifying previous steps.

5.5 Checking Invalid Expressions

During the simulation, KANAL checks the truth/falsity of many assertions, especially for the precondition tests and the expected effect tests. Whenever there are objects tested but undefined, KANAL reports the problem of accessing undefined objects (or invalid expressions).

5.6 Checking Loops

Loops are not necessarily a problem in process models. For example, the replication of DNA can be repeated multiple times. However, they can be unintended repetitions especially when the user defines many ordering constraints across steps. KANAL provides a warning for such cases to let the user check if the loops are in fact intended.

5.7 Checking Disjunctive Branches

When there are disjunctive branches in a model, the user may not notice that some of the combinations of alternatives should in fact not be possible. KANAL exposes different branches in the models by showing different alternative combinations of substeps. As in the case of loops, disjunctive branches are not necessarily a problem and KANAL simply informs the user about them.

5.8 Checking Causal Links

After the simulation, KANAL computes the interdependencies between the steps based on how some steps generated effects that satisfied the preconditions of some other steps. For example, synthesizing viral protein needed for replication enables the replication step. These causal links may not have been explicitly indicated by the user. KANAL informs users when it notices causal links in order to help them validate the model. It also informs users when they specified a temporal constraint and there does not seem to be a causal link between the steps to justify the ordering.

6 Interaction among problems and fixes

This section shows some examples of the kinds of errors detected and the fixes proposed by KANAL from a simplified Lambda virus invasion model shown in Figure 3. (In the model, first a lambda virus moves next to a cell (Arrive), and then it enters into the cell (Enter). The lambda DNA forms a circle (Circularize), and then either becomes integrated with

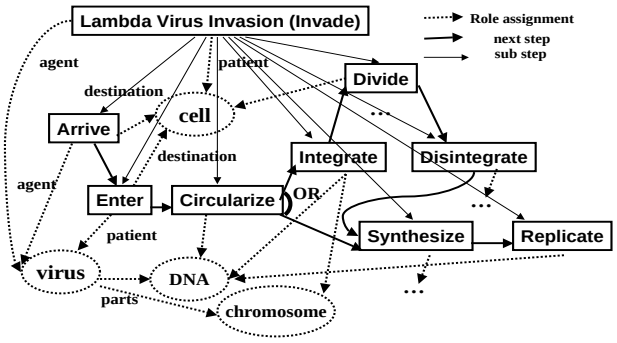


Figure 3: A simplified Lambda virus invasion model.

the host chromosome (Integrate) or starts synthesizing viral protein needed for the formation of new viruses (Synthesize). When it gets integrated, it lies dormant in the chromosome during cell divisions (Divide). An environmental change can induce the genome to leave the host chromosome (DisIntegrate) and to start synthesizing viral protein and subsequent DNA replication (Replicate) [Alberts *et al.*, 1998].)

- inappropriate execution of a step

  When the agent (virus) of the Arrive step is not specified, KANAL detects inappropriate execution of the step since its delete-list contains assertions that are invalid (the location of a Thing instead of the location of the virus). In such cases, KANAL proposes to either to modify the assignments of the Arrive step or to add some steps before Arrive so that they can assert the location of the Thing. The first fix directly points to the deleted link, and we call it a *direct fix*. The latter one will eventually lead the user to the problem because it mentions the location but does not point to it directly. We call it an *indirect fix*.

- unachieved preconditions

  In the above case (missing agent of the Arrive step), the precondition of the Enter step that follows (location of the virus should be near the cell) may also fail because of the failure of the Arrive step. KANAL detects this unachieved precondition and proposes multiple ways of fixing the problem: (1) modify the Arrive step so that it can change the location of the virus, (2) add a new Move or Arrive step to achieve the precondition, or (3) modify the Enter step to have different preconditions. Note that in this case, the same fix (i.e., modifying the Arrive step) is suggested for solving two different errors (inappropriate execution of Arrive and unachieved precondition of Enter).

  Also, the same set of fixes may resolve two different errors. For example, if the assignment of the patient (virus) of Enter is missing, its precondition (the patient should be near the destination) fails. KANAL produces an error message and proposes similar fixes as in the above case.

- failed expected effects

One of the expected effects was that the the virus should be inside the cell after an execution of the scenario. When the assignment of destination (the cell) of the Enter step is missing, the effect (the virus enters into the cell) cannot be achieved.

- missing ordering constraints

  When ordering between steps are missing (e.g., the link between the Arrive step and the Enter step is not specified), KANAL interrupts the simulation and reports an error. It proposes to add or modify temporal links between the steps so that the simulator can execute the unreached steps. When there is any ambiguity in the ordering, it generates a warning, asking for explicit ordering constraints among the steps.

In addition to the above, KANAL reports other results from the simulation:

- causal links

  There were three causal links reported from the scenario: (1) Arrive enabled Enter by achieving the location the virus being near the cell, (2) Integrate enabled Disintegrate by achieving the DNA being integrated with the chromosome, and (3) Synthesize enabled Replicate by providing the viral protein needed for the replication.

- disjunctive links

  The disjunctive branches stemming from Circularize were shown to the user as two alternative paths.

- simulated paths

  There were two simulated paths because of the disjunctive branches: (Arrive → Enter → Circularize → Integrate → Divide → DisIntegrate → Synthesize → Replicate) and (Arrive → Enter → Circularize → Synthesize → Replicate).

Notice that KANAL can detect the same error in multiple ways since one abnormality can lead to another. For example, missing an ordering constraint can make some steps unreached during the simulation, which can also lead to failed expected effects because of the unexecuted (unreached) steps. Since whenever there are unreached steps there tends to be failed expected effects, users may want to focus on fixing problems about the unreached steps first. The same case holds for failed preconditions and unreached steps because failed preconditions interrupt simulation, leading to unreached steps. To help avoid confusion, KANAL can selectively present fixes so that the user can concentrate on the actual source of the problem. For the case of failed preconditions and unreached steps, KANAL presents the fixes for the failed preconditions first, but lets users check other fixes if they want.

7 Preliminary Evaluation

KANAL is being integrated with concept composition, explanation tools and their interfaces in the end-to-end system mentioned above, and we are planning to perform an extensive user evaluation of this integrated system. The preliminary evaluation presented here focuses on how useful KANAL is in itself as a module to detect and fix errors.

Results w/o 1 link	Virus Invasion	Lambda Virus Invasion	Check Drain Monitor	total
# of test cases	19	28	9	56
# of errors	19	28	9	56
# of errors detected	18	28	9	55
# of errors with direct fixes	16	26	8	50
total # of fixes	82	139	23	230
- # of direct fixes	17	31	8	56
Avg. # of fixes proposed	4.56	4.96	2.56	4.18 (avg)

Results w/o 2 links	Virus Invasion	Lambda Virus Invasion	Check Drain Monitor	total
# of test cases	10	10	10	30
# of errors	20	20	20	60
# of errors detected	13	14	15	42
# of errors with direct fixes	13	13	13	39
# of fixes proposed	76	28	23	127
- # of direct fixes	17	12	13	42
Avg. # of fixes proposed	5.85	2	1.53	3.02 (avg)

Results w/o 3 links	Virus Invasion	Lambda Virus Invasion	Check Drain Monitor	total
# of test cases	10	10	10	30
# of errors	30	30	30	90
# of errors detected	13	16	18	47
# of errors with direct fixes	12	16	15	43
# of fixes proposed	62	58	54	174
- # of direct fixes	14	16	15	45
Avg. # of fixes proposed	4.77	3.63	3	3.70 (avg)

Table 1: KANAL checks for Process Models.

To evaluate KANAL's help in detecting and fixing errors, we used three process models: a virus invasion process, a Lambda virus invasion, and a Check-Condensate-Drain-Motor procedure from a High Pressure Air Compressor (HPAC) domain which has been also used for acquiring process models (lessons) for intelligent tutoring systems [Scholer *et al.*, 2000]. The first and last ones were written by other researchers.

We evaluated how KANAL could help users with one important kind of error: if they forget to specify one, two, or three links or role assignments. For example, to specify the Virus Invasion model the user would need to make a total of 19 links and assignments if no errors are made. The test cases were generated by taking the original correct process models and randomly deleting a subset of the links or assignments that users would need to make.

Table 1 shows the results from our preliminary evaluation. The first rows in each table show the number of cases tested and the second rows show the total number of errors in the test cases. The third row shows how many of those errors were detected by KANAL. KANAL was able to detect most of the errors when there is only one error (55 of 56). KANAL misses one case in the Lambda virus invasion

model because its Invade component has Container as its patient which should in fact be a Cell (a more special concept than Container), but there was no explicit violation in any of the checks KANAL performs. To be able to detect such problems we may need to examine slots tested in the model and check if they in fact belong to the concept. For example, Cells contain cytoplasm but not any Containers do in general.

KANAL missed some errors when more than one link were deleted (42 among 60 without 2 links, and 47 among 90 without 3 links). This is to be expected, since some errors interrupt the simulation, and other errors cannot be detected unless further steps are simulated, as described in the previous section.

The number of errors which had direct fixes are shown in the fourth rows in the tables. There were a few cases where KANAL detected errors but was not able to provide direct fixes. For example a step's role can refer to a deleted slot of another step, such as when the agent of the Enter step refers to the agent of the Invade step. If the Enter step failed because of the missing agent of the Invade step, then the failed step is different from the step with missing links, making it harder to find the sources of the problems. We are planning to examine how we can follow such links among the steps to trace back to the original sources. Some other cases of lack of direct fixes happened when there are multiple errors at the same time because some errors are hidden as described above. KANAL's selective presentation of fixes helps, and we expect that fixing one problem at a time may be easier for end users.

The numbers of fixes shown in the fifth rows of the tables are based on the selected fixes described above. Direct fixes should be more useful than indirect fixes in general. The number of direct fixes are shown in the sixth rows in the tables. For most of the errors detected, KANAL was able to provide at least one direct fix.

In summary, our preliminary evaluations show:

- KANAL virtually always (115 of 116 test cases) detected an error and made suggestions to the user. The one case that KANAL missed was a process model that was perfectly consistent although it was overgeneral, which is a problem that could only be noticed by a user.
- Detecting an error impaired detecting others since posterior steps will not be executed in the simulation. Although KANAL detected 98%, 70%, and 52% of the errors in the case where one, two, and three links and assignments were missing, it always detected at least one error in each of the process models that had more than one error. Once an error was fixed, the next error was always found by KANAL (direct fixes).
- For 91.6% of the errors detected, KANAL's fixes pointed directly to the source of the errors.

8 Conclusions

As we mentioned, we are planning to perform an extended evaluation with end users (biologists) when KANAL is integrated with the end-to-end system described earlier. In doing so, we will also be able to test the usefulness of KANAL with different types of errors than the ones we show here, including selecting wrong components in the knowledge base.

There have been a lot of verification and validation techniques developed in software engineering [Wallace *et al.*, 1996; Basili, 1987]. Although many of them are not directly applicable, there are many common issues in building process information, including efficiency, maintenance, cost, reuse, etc. We are planning to examine useful techniques developed for such issues.

KANAL is built for concept composition where components in the knowledge base are assumed not to have any errors. However, in other environments, such as in acquiring procedural knowledge for intelligent tutoring systems, we cannot expect that the models of actions will always be correct. Often, incomplete operators are used to model procedures and they are refined based on instructor input or through autonomous learning by experimentation. We believe that KANAL will be also useful in such environments.

Acknowledgments

We would like to thank Bruce Porter and Peter Clark for their help in integrating KANAL with the KM simulator and for providing the virus invasion scenario, and Andrew Scholer for his help with the HPAC process model and with the simulator of the tutoring system mentioned. We would also like to thank Vinay Chaudhri, Mabry Tyson, and Jerome Thomere for their comments and feedback on this work. Kevin Knight provided very helpful comments on earlier drafts. This research was funded by the DARPA Rapid Knowledge Formation (RKF) program with subcontract number 34-000-145 to SRI International under contract number N66001-00-C-8018.

References

[Alberts *et al.*, 1998] Alberts, B., Bray, D., Johnson, A., Lewis, J., Raff, M., Roberts, K. & Walter, P. *Essential Cell Biology: An Introduction to the Molecular Biology of the Cell*. Garland Publishing Inc, 1998.

[Basili, 1987] Basili, V. & Selby R. Comparing the effectiveness of software testing strategies. In *IEEE Transactions on Software Engineering*, 13(12), 1987.

[Chien, 1998] Chien, S. Static and completion analysis for knowledge acquisition, validation and maintenance of planning knowledge bases. In *International Journal of Human-Computer Studies*, 48, pp. 499–519, 1998.

[Clark & Porter, 1997] Clark, P. & Porter, B. Building concept representations from reusable components. In *Proceedings of AAAI-97*, pp. 369-376, 1997.

[Clark & Porter, 2000] Clark, P. & Porter, B. The knowledge machine. In *http://www.cs.utexas.edu/users/mfkb/km.html*, 2000.

[Ghallab *et al.*, 1998] Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D. & Wilkins, D. PDDL - the planning domain definition language. Technical report, Yale University. Available at http://www.cs.yale.edu/pub/mcdermott/software/pddl.tar.gz.

[Huffman & Laird, 1995] Huffman, S. & Laird, J. Flexibly instructable agents. In *Journal of Artificial Intelligence Research*, 3:271–324, 1995.

[Kambhampati, Knoblock, & Yang, 1995] Kambhampati, S.; Knoblock, C.; and Yang, Q. 1995. Planing as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning. *Artificial Intelligence* 76:167–238.

[Kim & Gil, 1999] Kim, J. & Gil, Y. Deriving expectations to guide knowledge base creation. In *Proceedings of AAAI-99*, pp. 235–241, 1999.

[Kim & Gil, 2000] Kim, J. & Gil, Y. Acquiring problem-solving knowledge from end users: Putting interdependency models to the test. In *Proceedings of AAAI-2000*, pp. 223–229, 2000.

[McGuinness *et al.*, 2000] McGuinness, D., Fikes, R., Rice, J. & Wilder, S. An Environment for Merging and Testing Large Ontologies. In *Proceedings of KR-2000*, 2000

[Myers, 1996] Myers, K. Strategic advice for hierarchical planners. In *Proceedings of KR-96*, 1996.

[O'Keefe & O'Leary, 1994] O'Keefe, R. & O'Leary, D. Expert system verification and validation. In *Expert Systems with Applications: An International Journal*, 6(1): 57-66.

[Preece & Shinghal, 1994] Preece, A. & Shinghal, R. Foundation and application of knowledge base verification. In *International Journal of Intelligent Systems*, 9(8): 683-702, 1994

[Sacerdoti, 1977] Sacerdoti, E. 1977. *A Structure for Plans and Behavior*. New York: Elsevier.

[Scholer *et al.*, 2000] Scholer, A., Rickel, J., Angros, R. & Johnson, L. Learning domain knowledge for teaching procedural tasks. In *AAAI-2000 Fall symposium on Learning How to Do Things*, 2000

[Swartout & Gil, 1995] Swartout, W. & Gil, Y. EXPECT: Explicit representations for flexible acquisition. In *Proceedings of KAW-95*, 1995.

[Tate, 1977] Tate, A. 1977. Generating project networks. In *International Joint Conference on Artificial Intelligence*.

[Tate, 1996] Tate, A. 1996. Representing plans as a set of constraints – the <i-n-ova> model. In Drabble, B., ed., *Proc. Third International Conference on Artificial Intelligence Planning Systems*. University of Edinburgh: AAAI Press. Available as Pointer.

[Tissot & Gruninger, 1999] Tissot, F., & Gruninger, M. NIST process specification language. Technical report, NIST.

[Wallace *et al.*, 1996] Wallace, D., Ippolito, L. & Cuthill B. Reference information for the software verification and validation Process. In *NIST Special Publication 500-234*, 1996.

[Wilkins, 1988] Wilkins, D. E. 1988. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann.

[Yang, 1990] Yang, Q. 1990. Formalizing planning knowledge for hierarchical planning. *Computational Intelligence* 6(1):12–24.

Integrating Expectations from Different Sources to Help End Users Acquire Procedural Knowledge

Jim Blythe
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292, USA
blythe@isi.edu

Abstract

Role-limiting approaches using explicit theories of problem-solving have been successful for acquiring knowledge from domain experts[1]. However most systems using this approach do not support acquiring procedural knowledge, only instance and type information. Approaches using interdependencies among different pieces of knowledge have been successful for acquiring procedural knowledge, but these approaches usually do not provide all the support that domain experts require. We show how the two approaches can be combined in such a way that each benefits from information provided by the other. We extend the role-limiting approach with a knowledge acquisition tool that dynamically generates questions for the user based on the problem solving method. This allows a more flexible interaction pattern. When users add knowledge, this tool generates expectations for the procedural knowledge that is to be added. When these procedures are refined, new expectations are created from interdependency models that in turn refine the information used by the system. The implemented KA tool provides broader support than previously implemented systems. Preliminary evaluations in a travel planning domain show that users who are not programmers can, with little training, specify executable procedural knowledge to customize an intelligent system.

1 Introduction

In order to be successful, deployed intelligent systems must be able to cope with changes in their task specification. They should allow users to make modifications to the system to control how tasks are performed and to specify new tasks within the general capabilities of the system. For example, consider a travel planning assistant that can locate flights and hotel reservations to help a user create an itinerary for some trip. Many such systems allow users to search for hotels by cost, hotel chain and distance to some location. However, users often have individual requirements, such as "prefer a direct flight unless it is double the price of the cheapest connecting flight" or "if the flight arrives late in the evening, the hotel should be near the airport, otherwise it should be near the meeting." These requirements often go beyond the initial abilities of the travel assistant tool, leaving the user to check them by hand and severely limiting the tool's usefulness.

The ability to define requirements like these in a way that can be integrated with the tool is therefore essential for it to meet a wide range of users' needs. The requirements, which were suggested by an independent user, are typical in that they do not require adding new sources of information to the tool, which can already compute distances and knows flight arrival times. Instead, they require processing the information to produce new criteria: in the first case for instance, finding the minimum value of the price for the set of all connecting flights and multiplying this value by 2. These are instances of *procedural* knowledge, rather than purely *factual* knowledge like the distances or costs. A tool that can incorporate new procedural knowledge like this from users can have a range of applicability that goes well beyond that originally envisaged by the developer.

However, it is difficult for users who are not programmers to add procedural knowledge to systems. In the next section we discuss some of the challenges that users face in more detail. Some KA approaches use *expectations* of the entered knowledge to aid users [Kim & Gil, 1999]. Expectations are beliefs about the knowledge that the user is currently entering that can be used to constrain the possibilities for what can be entered next. For example, expectations can govern the return type of a function that the user is entering, or its general purpose within the system. They can be used both to interpret and check knowledge as it is entered, to provide feedback or to help a user enter correct knowledge.

Another common direction of work in knowledge acquisition (KA) aims to support users through explicit, domain-independent theories of the tool's problem-solving process, often called *problem-solving methods* (PSMs) [Breuker & de Velde, 1994; Eriksson *et al.*, 1995]. These theories can encourage re-use across different applications and the structured development of intelligent systems as well as providing a guide for knowledge acquisition from experts. They can be

[1]I gratefully acknowledge support from DARPA grants F30602-00-2-0513, as part of the Active Templates program, and F30602-97-1-0195, as part of the High Performance Knowledge Bases program

used to structure the KA session for the user and provide context for the knowledge that is acquired. However, most KA approaches that use problem-solving methods focus on assisting knowledge engineers rather than domain experts [Fensel & Benjamins, 1998; Fensel & Motta, 1998].

Other KA tools such as SALT [Marcus & McDermott, 1989] take a *role-limiting* approach, allowing domain experts to provide domain-specific knowledge that fills certain roles within a PSM. However, most of these have been used to acquire instance-type information only. Musen [Musen, 1992] argues that although role-limiting provides strong guidance for KA, it lacks the flexibility needed for constructing knowledge-based systems (KBS). The problem-solving structure of an application cannot always be defined in domain-independent terms, and a single problem-solving strategy may be too general to address the particulars of an application. Puerta et al. advocate using finer-grained PSMs from which a KBS can be constructed [Puerta *et al.*, 1992].

Gil and Melz [Gil & Melz, 1996] address this problem by encoding the PSM in a language that allows any part of the problem-solving knowledge to be inspected and changed by a user. In their approach, a partially completed KBS can be analyzed to find missing problem-solving knowledge that forms the roles to be filled. This is done as part of the *interdependency analysis* performed by EXPECT [Swartout & Gil, 1995; Kim & Gil, 1999], which looks at how both problem-solving knowledge and factual knowledge is used in the intelligent system. This work extended the role-limiting approach to acquire problem-solving knowledge and to determine the roles dynamically. However, Gil and Melz's tools were not adequate for end users. There are at least two reasons for this. First, there is no support for a structured interaction with the user as there is in tools like SALT. The knowledge roles, once generated, form an unstructured list of items to be added, and it can be difficult for the user to see where each missing piece of knowledge should fit into the new KBS. Second, the user must work directly with their procedure syntax to add problem-solving knowledge, which is not appropriate for end users.

One solution is to exploit knowledge from a variety of sources to guide the user through all the stages of adding procedural knowledge. We view all the KA tools mentioned above as providing different kinds of expectations on the knowledge to be entered, either from background theories in the form of the PSMs, or from interdependency analysis. This framework allows the tool to exploit the background theory from the PSM to help a user begin the process of adding knowledge, and also to exploit interdependencies to help a user refine an initial definition of a procedure into an executable one. In our implemented system, which is built on EXPECT, modules that use expectations from the two sources share information in the form of input-output characterizations of expected procedural knowledge. This sharing is mutually beneficial to both the background theory-based and interdependency-based approaches.

In the next section I discuss some of the challenges that users who are not programmers face in defining procedural knowledge. Next I describe the use of expectations in more detail and show how they are integrated in an implemented tool, called *Constable*. I then report on initial user experiments with Constable that demonstrate the value of the approach.

2 Why do users find it difficult to enter procedural knowledge?

There are several challenges that users face in defining procedural knowledge. Here I sketch how some of them can be addressed, and highlight the role played by expectations.

Users do not know where to start. Adding a new capability to an intelligent system may require adding several related pieces of knowledge, in a form recognizable by the system. Simply beginning this process can be difficult even for an experienced programmer who does not know the system well. Expectations based on the **problem-solving method** can help identify the purpose and the initial structure of new procedural knowledge [Eriksson *et al.*, 1995].

Users do not know formal languages. A structured English editor allows users to modify English paraphrases of a procedure's formal representation [Blythe & Ramachandran, 1999]. Users can select a fragment of the paraphrase and choose from a list of suggested replacements for the fragment, which are automatically generated based on **expectations from method analysis**. This approach hides the internal procedure syntax while avoiding the challenge of full natural language processing.

Users may not know whether the added knowledge is free of syntax errors. The new knowledge may have errors in formal syntax (*e.g.* an "if" statement with no condition) or it may have type errors (*e.g.* trying to multiply the result of a subprocedure that returns a hotel). Since users create procedures by choosing replacements from a list, the editor can effectively eliminate some syntax errors. Others can be detected to generate a warning. If a procedure fragment is selected that contributes to a syntax error, some of the suggested replacements are formulated to fix the error, further helping the user.

Users may not know whether the added knowledge is correct. The procedure may be correct from a formal standpoint but not achieve the desired result. Constable tests new knowledge against examples as soon as it is entered to help find these problems.

It takes several steps to add new knowledge, so users can easily be lost. Users often do not realize and/or forget the side effects of the changes that require following up. Expectations from the PSM can be used to guide users through the initial steps in adding new knowledge. This is done through a script, as the next section shows. The approach is related to knowledge acquisition scripts [Tallis & Gil, 1999], though not as general.

This discussion of problems that users face indicates that providing help based on a combination of several different kinds of expectations may be key for non-programmers to add procedural knowledge. In the next two sections we show the help that can be given based on different kinds of expectations and describe how they are integrated in Constable. We begin by describing expectations from PSM task theories and then describe how the system makes use of expectations

derived from analyzing procedure interdependencies. In the following section we describe how the expectations are integrated by expressing expectations from background theories in terms of input-output type expectations.

3 Expectations from background theories

Expectations derived from the background theory in a problem-solving method are used to help clarify the purpose of the new procedural knowledge to be added by identifying its place in the PSM framework. Once this identification is made, initial templates are created for the new knowledge, which the user can refine until they perform the desired task. In this way, expectations from background theories help a user begin the process of defining new procedural knowledge to perform some task. In our approach, the background theory has two main components: (1) an ontology of concepts related to the task, and (2) generic procedural knowledge for performing each subtask.

This approach is general and can be applied to a wide range of generic tasks. In this paper we use an implemented problem-solving method for plan evaluation to illustrate the approach. Plan evaluation problems belong to a domain-independent problem class in which an agent, typically a human expert, judges alternative plans according to a number of criteria. The aim is usually to see which of the alternative plans is most suited for some task. In terms of a standard problem solving method library such as CommonKADS [Breuker & de Velde, 1994], it is a special case of assessment.

Each criterion for judging a plan is represented explicitly in this framework. Through experience with several intelligent systems for plan evaluation [Valente *et al.*, 1999], we have identified several patterns in the ways that the criteria are evaluated [Blythe & Gil, 1999]. These patterns are regularities that can be re-used across planning domains and provide guidance for knowledge acquisition. They are represented through an ontology of plan judgment criteria, called *critiques*, that is partially shown in figure 1. For example, `upper-bound` represents the class of critiques that can be evaluated by checking that some property of the plan has a value that is below a maximum value. Each class is identified with a pattern for evaluating a plan, implemented through generic procedural knowledge attached to the class. The PSM also includes concepts related to plans and the use of resources.

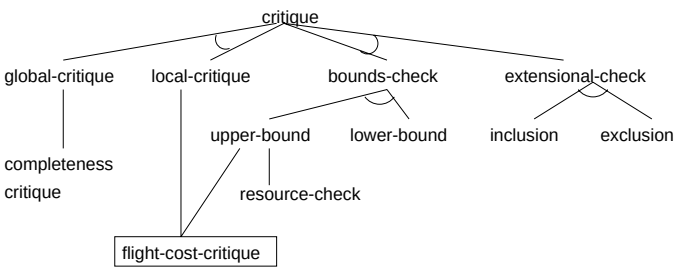


Figure 1: Different types of criteria for judging plans are part of the background theory of plan evaluation.

The second component of the task theory consists of generic procedural knowledge attached to some of the subtasks within the domain. In the plan evaluation domain, these subtasks are the generic critique types. For example, the following method says that a step satisfies an upper bound critique if and only if the actual value of the associated property is less than or equal to its maximum value. The tasks of estimating the actual and maximum values for the property are two methods that can be defined by the user with our tool.

capability: (determine-whether (obj (?thing is (inst-of thing)))
(satisfies (?bc is (inst-of upper-bound))))
result-type: (inst-of boolean))
method:
(check-that (obj (estimate (obj actual-value) (of ?bc) (for ?thing)))
(is-less-than-or-equal-to (estimate (obj maximum-allowed-value)
(of ?bc) (for ?thing))))

3.1 Using background theories in Constable

The background theory can be used to create a working plan evaluation system for a particular domain by defining domain-specific critique classes within the ontology and adding the procedures needed complete each critique's definition. In the travel planning domain, for example, plans are itineraries for travel and the steps in plans represent reservations of flights, hotels and rental cars. One possible critique checks that no flight costs more than $500. This is implemented by defining the critique `flight-cost` as a subclass of both `local-critique` and `upper-bound`. The procedures for those classes are then used to evaluate the new critique, resulting in a check that the actual amount of `flight-cost` for each step is less than or equal to the maximum amount. The user completes the definition by defining methods to compute the actual amount (by retrieving the cost of the flight) and the maximum allowed amount ($500).

Figure 2 shows the main window through which a user defines the `flight-cost` critique in Constable. The tool presents questions of two kinds: those aimed at classifying the critique in the ontology, *e.g.* questions 2, 5 and 7, and those allowing the user to refine default procedural knowledge, which begin with the phrase "show me how to...". The questions are attached to the critique classes in the ontology and the tool asks them as it tries to classify the new critique. For instance, question 5, "Warn if flight-cost is too large?" is used to classify the critique as an upper bound. Once a positive classification is made, the tool gives the user an option to refine the procedural knowledge attached to the class. The generic procedural knowledge for this class include a default for "estimate the maximum allowed value of ..", so question 6 allows the user to refine this default for the flight cost.

The use of background theories to classify new knowledge and define default procedural knowledge helps to solve the first problem that users face in creating procedural knowledge: how to get started. The tool begins by asking questions about the nature of the knowledge to be added, and the default procedures are guaranteed to be applicable within the system. The task theory is also used to break the new knowledge into manageable pieces, an important step that is hard for users who are not programmers. However, refining the methods to compute actual values and maximum allowed values for flight costs, for example, can still be a daunting task requiring the tool to offer more assistance. Expectations based on

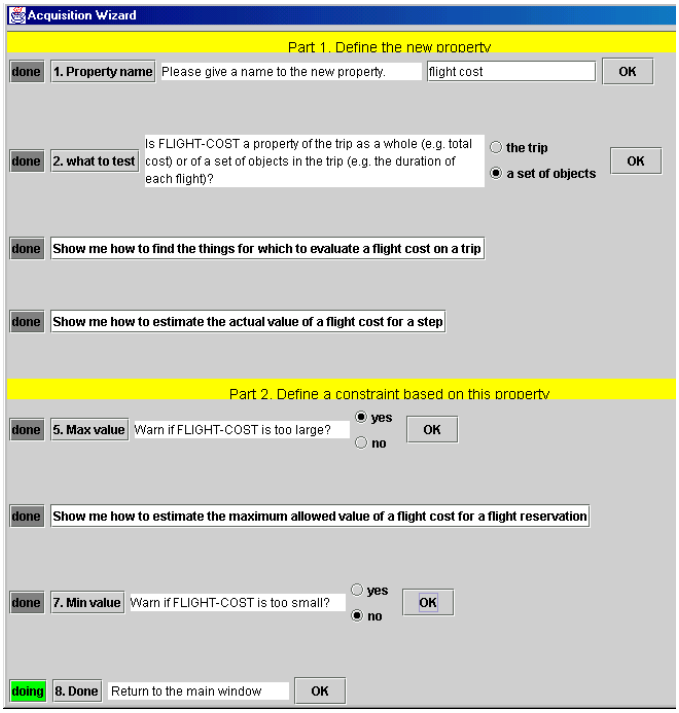


Figure 2: Constable's main window for defining a critique includes questions that classify the new critique within the ontology and questions that refine the generic procedural knowledge associated with the classification.

interdependency analysis are one source of this assistance.

4 Expectations from interdependencies

The expectations described in the last section come from background theories of tasks in problem-solving methods. Here we consider expectations that come from comparing the new procedural knowledge with the procedure syntax and with existing procedural or factual knowledge, which we refer to as *interdependency expectations*. Some examples of syntax-based expectations are that variables that are used in the procedure body are defined and that the rules of syntactic constructs such as "*if ... then ... else ...*" are observed. Some examples of expectations generated by comparing the new knowledge with existing procedural and factual knowledge are that relations and objects used are defined in the knowledge base, that other procedural knowledge that is used is defined, and that the relations and procedural knowledge used will return information of an appropriate type for the way it is being used (*e.g.* the procedure does not try to multiply a hotel by a number). These expectations are derived from interdependency models [Kim & Gil, 1999].

Figure 3 shows Constable's editor for procedural knowledge being used to define how to compute the maximum allowed value of the cost of a flight reservation. The purpose and body of the procedure are automatically paraphrased in English [Blythe & Ramachandran, 1999]. When the user selects part of the procedure to change, in this case "*Pittsburgh*", the editor suggests replacements in the lower panel, also automatically paraphrased. The suggestions are generated by analyzing the current knowledge base for possible terms and grouping them. Examples of groups are procedures or relations that could be applied to the current term or that have the same type. This approach hides the formal syntax while avoiding the challenge of full natural language processing.

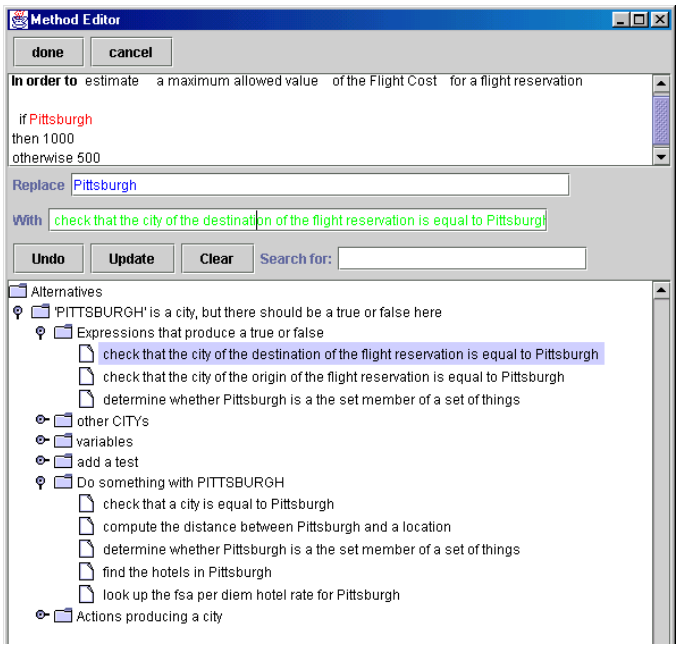


Figure 3: Constable's editor being used to define how to compute the maximum allowed value of the cost of a flight reservation.

We distinguish two ways that a KA tool can use interdependency expectations to help a user define procedural knowledge, called hard and soft expectations. For a *hard expectation*, the tool does not allow procedural information that violates the expectation to be expressed. For example, the only way for a user to introduce an "if" construct into a procedure is to select one from the suggested replacements, so the user can never express a procedure that violates the basic syntax of the construct. All syntax expectations are hard expectations in this editor.

For a *soft expectation*, the tool will allow a user to define procedures that violate the expectation, but will provide warnings, and possibly remedies, for the violation. For example the method in Figure 3 currently violates a type expectation. To compute the maximum allowed cost for a flight, the user specifies "if Pittsburgh then 1000, otherwise 500", but in the formal syntax, the keyword "if" must be followed by code that returns a boolean value. The editor shows an error message in the lower panel and, when "Pittsburgh" is selected to be replaced, suggests expressions that include "Pittsburgh" but will produce a boolean value, including "check that the destination city is equal to Pittsburgh" and "check that the origin city is equal to Pittsburgh". These expressions are produced by an anytime search algorithm. The suggestions

also include expressions that will not fix the violation, such as "compute the distance between Pittsburgh and a location".

Choosing to make an expectation hard or soft can impact how easily a user can express a procedure through successive replacements and how easily a user can be confused by an intermediate form of the procedure. Empirically, users find it useful to build intermediate expressions that violate interdependency expectations, such as in the above example or by using undefined procedures. This allows users to concentrate on other issues before fixing the violations. These are soft expectations in Constable. However it is not so useful for users to violate syntactic expectations; these are hard in Constable. Different KA tools may choose different expectations to be hard or soft, and the optimal choice may also depend on the skill level of the user.

Almost all compilers find and report errors based on input-output expectations. Constable goes beyond this in suggesting modifications to the existing code that would address the error. The suggestions are found by a breadth-first search through the space of possible sentences, taking the user-selected fragment as a starting point. A node in the search space is a term in the formal syntax, and it is expanded by applying all known applicable relations or procedures. The search terminates when terms are reached that resolve the expectation violation, or match a user-typed search string. Since the search space can be infinite, the suggestion module times out after a short time.

Users who are not programmers can be daunted by the need for executable procedural knowledge to have precise syntax with subgoals returning correctly typed information. The use of the expectations described above, in both hard and soft form, can provide significant help.

5 Integrating expectations from different sources

The previous two sections described how both expectations from background theories and method-based expectations provide valuable help for users to define procedural knowledge. In combination, the two kinds of expectations can provide more assistance than the sum of their parts. This is because useful information about the emerging procedural knowledge can be exchanged between the two sources.

There are several examples of this information flow when Constable is used to create the flight cost critique, summarized in the window in Figure 2. As we described earlier, the questions in Figure 2 are generated through expectations in the background task theory. Some of the questions are to classify the new critique in the ontology and some are to refine the attached procedural knowledge.

For each of the procedures to be defined in questions 3, 4 and 6, type expectations are sent from the task theory to the method analysis module, which uses them to help guide the user. Constable uses the task theory to assign types to each input variable and a desired output type, and calls the procedure editor. For example, the method "estimate the maximum allowed value of ..." has a desired type *(inst-of number)*, so the method editor will warn the user and suggest remedies if the method defined does not produce a number.

Some of the type expectations for the default methods are refined in the method analyzer by considering their interdependencies. For example, the user defines a method "estimate the actual value of flight cost for a flight reservation" in Figure 2. The input variable for this method has type *(inst-of flight-reservation)*, but this cannot be inferred from the task theory alone: it comes from the result type of the method "find the things for which to evaluate a flight cost on a trip". This is defined by the user to return a set of flight reservations. When this definition is complete, EXPECT's interdependency model is re-generated. It shows that the subsequent methods should take a flight reservation as input, and this information is passed back to the task theory from the method analyzer.

Information from the method analyzer is also used by the task theory in classifying the critique. For instance, once the user completes the definition of a procedure to "estimate the actual value of flight cost..", the method analyzer notes that it produces a number. This information is used in the task theory to classify the critique as a *bounds-check* according to the ontology of Figure 1. Question 5, "Warn if flight cost is too large?", attempts to classify the critique as a *lower-bound*. If the method to compute the actual value produced a different type, for example an airline, this question will not be asked. Instead, the task theory will classify the critique as an extensional critique and ask whether there are any preferred values, to test whether the critique is a *positive-extensional-critique*.

Information is passed between the PSM task theory and the method analyzer in both directions as the user defines a critique. The method analyzer makes use of type expectations from the task theory to help the user. It also passes refined type information to the task theory, based on the new procedural information and on interdependency analysis. The task analyzer uses this information as it classifies the new task. This interplay between the expectations from different sources significantly improves the guidance that can be given the user as the new knowledge is defined.

6 Preliminary experiments

We performed a preliminary evaluation of the approach by evaluating the performance of six subjects, who were not programmers, at adding and modifying critiques in the travel planning domain. In training, users followed written instructions to modify three critiques and add one new critique using Constable. In testing, users were asked to add and modify as many critiques as possible from a list of six. Subjects took an average of one hour to complete the training phase and thirty minutes working on the test critiques.

The simplest training task was to change a constant representing the maximum value of the distance between the hotel used and a meeting location. In the most complex task, the maximum distance for the hotel was defined as twice the distance of the closest available hotel. The procedure is:

```
(multiply (obj (find (obj (spec-of minimum))
                     (of (compute (obj (spec-of distance))
                                  (between (find (obj (set-of (spec-of hotel)))
                                                 (in (r-city (r-hotel ?t)))))
                                  (and (r-meeting (r-trip ?t)))))))
          (by 2))
```

All subjects entered this definition correctly using the tool. Even though they were following instructions, it is unlikely that this success rate would have been achieved without using Constable. The test tasks had the same range of complexity as the training tasks.

Adding or modifying a critique may require a number of steps to be completed. To measure the partial performance in the test tasks we counted each correctly defined method as a step, and also counted classifying the new critique correctly in the ontology as a step. Table 1 shows the average number of steps completed in each the first four test tasks. Every subject was able to make modifications to constant maximum values. Four of the six subjects were able to define a hotel cost critique according to the definition "If the city of the hotel is Pittsburgh, then the maximum hotel cost is $70, otherwise it is $120". Defining this kind of critique is beyond the scope of tools that do not allow users to define procedural knowledge.

Task	steps completed / Total steps
Modify constant maximum value	1/1
Max value is conditional on city	0.66 / 1
Complex new upper bound	1.4 / 3
Complex new upper bound II	1 / 3

Table 1: Average number of steps completed by subjects for each of four test tasks.

In addition to the pre-defined critiques, subjects attempted to add their own critiques to the plan evaluation tool. These critiques were written down before subjects worked through the training or test cases, so that they would not affect their choices. No subject succeeded in completely adding these critiques. Of the 26 critiques that subjects described, 17 could in principle be added through Constable. The others could be identified in the critique ontology but used relations and concepts, such as "bed-and-breakfast", that the tool did not include.

One shortcoming of the current tool is that subjects were not able to easily see all the concepts and relations available in the domain. The editor shows only those that are related to the current procedure definition. While this is helpful when users are making local changes to a procedure, they also need a way to see all available information. Also, the short time-scale of the experiment limited the time that subjects could spend thinking about the critiques. We hope to build a tool that the subjects would use on a daily basis that includes Constable, and investigate its impact.

After completing the experiment, several subjects whose job includes travel planning volunteered more critiques that would be useful in their work. It is interesting to compare these with the critiques that were expressed before the experiment: they all entail more complex procedural reasoning than the earlier critiques, but can be expressed in the critique ontology. Examples include "if the flight arrives late in the evening, the hotel should be near the airport, otherwise it should be near the meeting", and "prefer a hotel that is close to each of two meeting locations".

7 Discussion

Adding procedural knowledge to intelligent systems is a challenging task, but one that is necessary for the system to be useful to a wide range of users. We presented a novel approach in which expectations from background theories and from interdependency analysis are integrated to guide the user through the KA process. As well as providing assistance across a larger subset of the KA task than previous systems, the approach is able to combine information from the two sources to provide help that is not otherwise possible. In general, procedural knowledge should be added in concert with factual knowledge such as new concepts, relations and instances. Constable includes tools for adding factual knowledge which were not described here for space reasons. Information on more of the tools, but with less detail on the use of expectations, can be found in [Blythe *et al.*, 2001].

We are currently working on a number of applications of Constable. One is a travel planning tool that retrieves flight, hotel and other information from the web and uses constraints on partial travel plans to help users assemble an itinerary. Integrating Constable will help users specify individual preferences to assemble itineraries. We also intend to apply Constable to acquire procedural knowledge for reasoning about the biochemistry of DNA.

References

[Blythe *et al.*, 2001] Blythe, J., Kim, J., Ramachandran, S., and Gil, Y. 2001. An integrated environment for knowledge acquisition. Best Paper, *Proc. International Conference on Intelligent User Interfaces.*

[Blythe & Gil, 1999] Blythe, J., and Gil, Y. 1999. A problem-solving method for plan evaluation and critiquing. In *Proc. Twelfth Knowledge Acquisition for Knowledge-Based Systems Workshop.*

[Blythe & Ramachandran, 1999] Blythe, J., and Ramachandran, S. 1999. Knowledge acquisition using an english-based method editor. In *Proc. Twelfth Knowledge Acquisition for Knowledge-Based Systems Workshop.*

[Breuker & de Velde, 1994] Breuker, J., and de Velde, W. V. 1994. *CommonKADS Library for Expertise Modelling: Reusable Problem Solving Components.*

[Eriksson *et al.*, 1995] Eriksson, H.; Shahar, Y.; Tu, S. W.; Puerta, A. R.; and Musen, M. A. 1995. Task modeling with reusable problem-solving methods. *Artificial Intelligence* 79:293–326.

[Fensel & Benjamins, 1998] Fensel, D., and Benjamins, V. R. 1998. Key issues for automated problem-solving methods reuse. In *Proc. the European Conference on Artificial Intelligence.*

[Fensel & Motta, 1998] Fensel, D., and Motta, E. 1998. Structure development of problem solving methods. In *Proc. Eleventh Knowledge Acquisition for Knowledge-Based Systems Workshop.*

[Gil & Melz, 1996] Gil, Y., and Melz, E. 1996. Explicit representations of problem-solving strategies to support

knowledge acquisition. In *Proc. Thirteenth National Conference on Artificial Intelligence*.

[Kim & Gil, 1999] Kim, J., and Gil, Y. 1999. Deriving expectations to guide knowledge-base creation. In *Proc. Sixteenth National Conference on Artificial Intelligence,* 235–241. AAAI Press.

[Marcus & McDermott, 1989] Marcus, S., and McDermott, J. 1989. Salt: A knowledge acquisition language for propose-and-revise systems. *Artificial Intelligence* 39:1–37.

[Musen, 1992] Musen, M. A. 1992. Overcoming the limitations of role-limiting methods. *Knowledge Acquisition* 4(2):165–170.

[Puerta *et al.*, 1992] Puerta, A. R.; Egar, J. W.; Tu, S.; and Musen, M. A. 1992. A multiple-method knowledge acquisition shell for the automatic generation of knowledge acquisition tools. *Knowledge Acquisition* 4(2):171–196.

[Swartout & Gil, 1995] Swartout, W. and Gil, Y. 1995. EXPECT: Explicit Representations for Flexible Acquisition. In *Proc. Ninth Knowledge Acquisition for Knowledge-Based Systems Workshop*.

[Tallis & Gil, 1999] Tallis, M., and Gil, Y. 1999. Designing scripts to guide users in modifying knowledge-based systems. In *Proc. Sixteenth National Conference on Artificial Intelligence*. AAAI Press.

[Valente *et al.*, 1999] Valente, A.; Blythe, J.; Gil, Y.; and Swartout, W. 1999. On the role of humans in enterprise control systems: the experience of inspect. In *Proc. of the JFACC Symposium on Advances in Enterprise Control.*

MACHINE LEARNING AND DATA MINING

REINFORCEMENT LEARNING

R-MAX – A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning

Ronen I. Brafman
Computer Science Department
Ben-Gurion University
Beer-Sheva, 84105 Israel
brafman@cs.bgu.ac.il

Moshe Tennenholtz *
Computer Science Department
Stanford University
Stanford, CA 94305
moshe@robotics.stanford.edu

Abstract

R-MAX is a simple model-based reinforcement learning algorithm which can attain near-optimal average reward in polynomial time. In R-MAX, the agent always maintains a complete, but possibly inaccurate model of its environment and acts based on the optimal policy derived from this model. The model is initialized in an optimistic fashion: all actions in all states return the maximal possible reward (hence the name). During execution, the model is updated based on the agent's observations. R-MAX improves upon several previous algorithms: (1) It is simpler and more general than Kearns and Singh's E^3 algorithm, covering zero-sum stochastic games. (2) It has a built-in mechanism for resolving the exploration vs. exploitation dilemma. (3) It formally justifies the "optimism under uncertainty" bias used in many RL algorithms. (4) It is much simpler and more general than Brafman and Tennenholtz's LSG algorithm for learning in single controller stochastic games. (5) It generalizes the algorithm by Monderer and Tennenholtz for learning in repeated games. (6) It is the only algorithm for near-optimal learning in repeated games known to be polynomial, providing a much simpler and more efficient alternative to previous algorithms by Banos and by Megiddo.

1 Introduction

Reinforcement learning has attracted the attention of researchers in AI and related fields for quite some time. Many reinforcement learning algorithms exist and for some of them convergence rates are known. However, Kearns and Singh's E^3 algorithm [Kearns and Singh, 1998] was the first provably near-optimal polynomial time algorithm for learning in Markov decision processes (MDPs). E^3 was extended later to handle single controller stochastic games (SCSGs) [Brafman and Tennenholtz, 2000] as well as structured MDPs [Kearns and Koller, 1999]. In E^3 the agent learns by updating a model of its environment using statistics it collects. This learning process continues as long as it can be done relatively efficiently. Once this is no longer the case, the agent uses its learned model to compute an optimal policy and follows it. The success of this approach rests on two important properties: the agent can determine online whether an efficient learning policy exists, and if such a policy does not exist, it is guaranteed that the optimal policy with respect to the learned model will be approximately optimal with respect to the real world.

The difficulty in generalizing E^3 to adverserial contexts, i.e., to different classes of games, stems from the adversary's ability to influence the probability of reaching different states. In a game, the agent does not control its adversary's choices, nor can it predict them with any accuracy. Therefore, it has difficulty predicting the outcome of its actions and whether or not they will lead to new information. Consequently, it is unlikely that an agent can explicitly choose between an exploration and an exploitation policy. For this reason, the only extension of E^3 to adverserial contexts used the restricted SCSG model in which the adversary influences the reward of a game only, and not its dynamics.

To overcome this problem, we suggest a different approach in which the agent never attempts to learn explicitly. Our agent always attempts to optimize its behavior, albeit with respect to a fictitious model in which optimal behavior often leads to learning. This model assumes that the reward the agent obtains in any situation it is not too familiar with, is the maximal possible reward – R_{max}. The optimal policy with respect to the agent's *fictitious* model has a very interesting and useful property with respect to the *real* model: either it performs optimally, or it leads to efficient learning. The agent does *not* know whether it is optimizing or learning efficiently, but it always does one or the other. Thus, the agent will always either exploit or explore efficiently, without knowing ahead of time which of the two will occur. Since there is only a polynomial number of parameters to learn, as long as learning is done efficiently we can ensure that the agent spends a polynomial number of steps exploring, and the rest of the time will be spent exploiting. Thus, the resulting algorithm may be said to use an *implicit* explore or exploit approach, as opposed to Kearns and Singh's *explicit* explore or exploit approach.

This learning algorithm, which we call R-MAX, is very simple to understand and to implement. The algorithm con-

*Permanent address: Faculty of Industrial Engineering and Management, Technion, Haifa 32000, Israel

verges in polynomial-time to a near-optimal solution. Moreover, R-MAX is described in the context of zero-sum stochastic game, a model that is more general than Markov Decision Processes. As a consequence, R-MAX is more general and more efficient than a number of previous results. It generalizes the results of Kearns and Singh 1998 to adverserial contexts and to situations where the agent considers a stochastic model of the environment inappropriate, opting for a nondeterministic model instead. R-MAX can handle more classes of stochastic games than the LSG algorithm [Brafman and Tennenholtz, 2000]. In addition, it attains a higher expected average reward than LSG. R-MAX also improves upon previous algorithms for learning in repeated games [Aumann and Maschler, 1995], such as Megiddo's [Megiddo, 1980] and Banos [Banos, 1968]. It is the only polynomial time algorithm for this class of games that we know of, and it is much simpler, too. Finally, R-MAX generalizes the results of Monderer and Tennenholtz [Monderer and Tennenholtz, 1997] to handle the general probabilistic maximin (safety level) decision criterion.

The approach taken by R-MAX is not new. It has been referred to as *the optimism in the face of uncertainty* heuristic, and was considered an ad-hoc, though useful, approach (e.g., see Section 2.2.1 in [Kaelbling *et al.*, 1996], where it appears under the heading "Ad-Hoc Techniques" and Section 2.7 in [Sutton and Barto, 1998] where this approach is called *optimistic initial values* and is referred to as a "simple trick that can be quite effective on stationary problems"). This optimistic bias has been used in a number of well-known reinforcement learning algorithms, e.g. Kaelbling's interval exploration method [Kaelbling, 1993], the exploration bonus in Dyna [Sutton, 1990], the curiosity-driven exploration of [Schmidhuber, 1991], and the exploration mechanism in prioritized sweeping [Moore and Atkenson, 1993]. More recently, Tadepalli and Ok [Tadepalli and Ok, 1998] presented a reinforcement learning algorithm that works in the context of the undiscounted average-reward model used in this paper. One variant of their algorithm, called AH-learning, is very similar to R-MAX. However, as we noted above, none of this work provides theoretical justification for this very natural bias. Thus, an additional contribution of this paper is a formal justification for the *optimism under uncertainty* bias.

The paper is organized as follows: in Section 2 we define the learning problem more precisely and the relevant parameters. In Section 3 we describe the R-MAX algorithm. In Section 4 we sketch the proof that it yields near-optimal reward in polynomial time. We conclude in Section 5. Because of length limitations, proofs are not included. A full version of this work appears at `www.cs.bgu.ac.il/~brafman/sg01.ps`.

2 Preliminaries

We present R-MAX in the context of a model that is called a *stochastic game*. This model is more general than a Markov decision process because it does not necessarily assume that the environment acts stochastically (although it can). In what follows we define the basic model, describe the set of assumptions under which our algorithm operates, and define the parameters influencing its running time.

2.1 Stochastic Games

A game is a model of multi-agent interaction. In a game, we have a set of players, each of whom chooses some action to perform from a given set of actions. As a result of the players' combined choices, some outcome is obtained which is described numerically in the form of a payoff vector, i.e., a vector of values, one for each of the players. We concentrate on two-player, fixed-sum games (i.e., games in which the sum of values in the payoff vector is constant). We refer to the player under our control as *the agent*, whereas the other player will be called *the adversary*.

A common description of a game is as a matrix. This is called a game in *strategic form*. The rows of the matrix correspond to the agent's actions and the columns correspond to the adversary's actions. The entry in row i and column j in the game matrix contains the rewards obtained by the agent and the adversary if the agent plays his i^{th} action and the adversary plays his j^{th} action. We make the simplifying assumption that the size of the action set of both the agent and the adversary is identical. However, an extension to sets of different sizes is trivial.

In a *stochastic game* (SG) the players play a (possibly infinite) sequence of standard games from some given set of games. After playing each game, the players receive the appropriate payoff, as dictated by that game's matrix, and move to a new game. The identity of this new game depends, stochastically, on the previous game and on the players' actions in that previous game. Formally:

Definition 1 *A fixed-sum, two player, stochastic-game [SG] M on states $S = \{1, \ldots, N\}$, and actions $A = \{a_1, \ldots, a_k\}$, consists of:*

- **Stage Games:** *each state $s \in S$ is associated with a two-player, fixed-sum game in strategic form, where the action set of each player is A. We use R^i to denote the reward matrix associated with stage-game i.*
- **Probabilistic Transition Function:** *$P_M(s, t, a, a')$ is the probability of a transition from state s to state t given that the first player (the* agent*) plays a and the second player (the* adversary*) plays a'.*

An SG is similar to an MDP. In both models, actions lead to transitions between states of the world. The main difference is that in an MDP the transition depends on the action of a single agent whereas in an SG the transition depends on a joint-action of the agent and the adversary. In addition, in an SG, the reward obtained by the agent for performing an action depends on its action *and* the action of the adversary. To model this, we associate a game with every state. Therefore, we shall use the terms *state* and *game* interchangeably.

Stochastic games are useful not only in multi-agent contexts. They can be used instead of MDPs when we do not wish to model the environment (or certain aspects of it) stochastically. In that case, we can view the environment as an agent that can choose among different alternatives, without assuming that its choice is based on some probability distribution. This leads to behavior maximizing the worst-case

scenario. In addition, the adversaries that the agent meets in each of the stage-games could be different entities.

R-MAX is formulated as an algorithm for learning in Stochastic Games. However, it is immediately applicable to fixed-sum repeated games and to MDPs because both of these models are degenerate forms of SGs. A repeated game is an SG with a single state and an MDP is an SG in which the adversary has a single action at each state.

For ease of exposition we normalize both players' payoffs in each stage game to be non-negative reals between 0 and some constant R_{max}. We also take the number of actions to be constant. The set of possible histories of length t is $(S \times A^2)^t \times S$, and the set of possible histories, H, is the union of the sets of possible histories for all $t \geq 0$, where the set of possible histories of length 0 is S.

Given an SG, a policy for the agent is a mapping from H to the set of possible probability distributions over A. Hence, a policy determines the probability of choosing each particular action for each possible history.

We define the *value* of a policy using the *average expected reward criterion* as follows: Given an SG M and a natural number T, we denote the expected T-step undiscounted average reward of a policy π when the adversary follows a policy ρ, and where both π and ρ are executed starting from a state $s \in S$, by $U_M(s, \pi, \rho, T)$ (we omit subscripts denoting the SG when this causes no confusion). Let $U_M(s, \pi, T) = \min_{\rho \text{ is a policy}} U_M(s, \pi, \rho, T)$ denote the value that a policy π can guarantee in T steps starting from s. We define $U_M(s, \pi) = \liminf_{T \to \infty} U_M(s, \pi, T)$. Finally, we define $U_M(\pi) = \min_{s \in S} U_M(s, \pi)$.[1]

2.2 Assumptions, Complexity, and Optimality

In what follows, we make two central assumptions. First, we assume that the agent always recognizes the identity of the state (or stage-game) it reached (but not its associated payoffs and transition probabilities) and that after playing a game, it knows what actions were taken by its adversary and what payoffs were obtained. Second, we assume that the maximal possible reward R_{max} is known ahead of time. This latter assumption can be removed.[2]

Next, we wish to discuss the central parameter in the analysis of the complexity of R-MAX – the *mixing time*, first identified by Kearns and Singh 1998. Kearns and Singh argue that it is unreasonable to refer to the efficiency of learning algorithms without referring to the efficiency of convergence to a desired value. They defined the *ϵ-return mixing time* of a policy π to be the smallest value of T after which π *guarantees* an expected payoff of at least $U(\pi) - \epsilon$. In our case, we have to take into account the existence of an adversary. Therefore, we adjust this definition slightly as follows: a policy π belongs to the set $\Pi(\epsilon, T)$ of policies whose ϵ-return mixing time is at most T, if for any starting state s and for any adversary behavior ρ, we have that $U(s, \pi, \rho, T) > U(s, \pi) - \epsilon$.

That is, no matter what the initial state is and what the adversary does, on the average, we have to apply policy π no more than T steps before our average accumulated reward is sufficiently close to the value of π for the state s. Notice that this means that an agent with perfect information about the nature of the games and the transition function will require at least T steps, on the average, to obtain an optimal value using an optimal policy π whose ϵ-return mixing time is T. Clearly, one cannot expect an agent lacking this information to perform better.

We denote by $Opt(\Pi(\epsilon, T))$ the optimal expected undiscounted average return from among the policies in $\Pi(\epsilon, T)$. When looking for an optimal policy (with respect to policies that ϵ-mix in time T, for a given $\epsilon > 0$), we will be interested in approaching this value in time polynomial in T, in $1/\epsilon$, in $1/\delta$ (where δ is the failure probability), and in the size of the description of the game.

The reader may have noticed that we defined $U_M(\pi)$ as $\min_{s \in S} U_M(s, \pi)$. This choice may appear to make the learning task too easy. For instance, one may ask why shouldn't we try to attain the maximal value over all possible states, or at least the value of our initial state? We claim that the above is the only reasonable choice, and that it leads to results that are as strong as previous algorithms.

To understand this point, consider the following situation: we start learning at some state s whose value is very high. However, if we do not execute the action a in s, we reach some state s' that has a very low value. A learning algorithm without any prior knowledge cannot be expected to immediately guess that a should be done in s. In fact, without such prior knowledge, to conclude that a is appropriate it must compare its outcome to that of the other actions in s. Thus, one can expect an agent to learn a near-optimal policy only if the agent can visit state s sufficiently many times to learn about the consequences of different options in s. In a finite SG, there will be some set of states that we can sample sufficiently many times, and it is w.r.t. these states that we can learn how to behave.

In fact, it probably makes sense to restrict our attention to a subset of the states such that from each state in this set it is not too hard to get to any other state. In the context of MDPs, Kearns and Singh refer to this as the *ergodicity* assumption. In the context of SGs, Hoffman and Karp 1966 refer to this as the *irreducibility* assumption. An SG is said to be *irreducible* if the Markov-chain obtained by fixing any two (pure) stationary strategies for each of the players is irreducible (i.e., each state is reachable from each other state). In the special case of an MDP, irreducibility is precisely the ergodicity property used by Kearns and Singh in their analysis of E^3.

Irreducible SGs have a number of nice properties, as shown by [Hoffman and Karp, 1966]. First, the maximal long-term average reward is independent of the starting state, implying that $\min_{s \in S} U_M(s, \pi) = \max_{s \in S} U_M(s, \pi)$ for an optimal policy π. Second, this optimal value can be obtained by a stationary policy (i.e., one that depends on the current stage-game only). We believe that our results are primarily interesting in this class of games.

[1] We discuss this choice below.

[2] We would need to run the algorithm repeatedly for increasing values of R_{max}. The resulting algorithm remains polynomial in the relevant parameters.

3 The R-MAX algorithm

Recall that we consider a stochastic game M consisting of a set $S = \{G_1, \ldots, G_N\}$ of stage-games in each of which both the agent and the adversary have a set $A = \{a_1, \ldots, a_k\}$ of possible actions. We associate a reward matrix R^i with each game, and we use $R^i_{m,l}$ to denote a pair consisting of the reward obtained by the agent and the adversary after playing actions a_m and a_l in game G_i, respectively. In addition, we have a probabilistic transition function, P_M, such that $P_M(s, t, a, a')$ is the probability of making a transition from G_s to G_t given that the agent played a and the adversary played a'. It is convenient to think of $P_M(i, \cdot, a, a')$ as a function associated with the entry (a, a') in the stage-game G_i. This way, all model parameters, both rewards and transitions, are associated with joint actions of a particular game. Let $\epsilon > 0$. For ease of exposition, we assume throughout most of the analysis that the ϵ-return mixing time of the optimal policy, T, is known. Later, we show how this assumption can be relaxed.

The R-MAX algorithm is defined as follows:

Initialize: Construct the following model M' consisting of $N + 1$ stage-games, $\{G_0, G_1, \ldots, G_N\}$, and k actions, $\{a_1, \ldots, a_k\}$. Here, $G_1, \ldots, G_N$ correspond to the real games, $\{a_1, \ldots, a_k\}$ correspond to the real actions, and G_0 is an additional fictitious game. Initialize all game matrices to have $(R_{max}, 0)$ in all entries.[3] Initialize $P_M(G_i, G_0, a, a') = 1$ for all $i = 0, \ldots, N$ and for all actions a, a'.

In addition, maintain the following information for each entry in each game $G_1, \ldots, G_N$: (1) a boolean value *known/unknown*, initialized to *unknown*; (2) the states reached by playing the joint action corresponding to this entry (and how many times); (3) the reward obtained (by both players) when playing the joint action corresponding to this entry. Items 2 and 3 are initially empty.

Repeat:

Compute and Act: Compute an optimal T-step policy for the current state, and execute it for T-steps or until a new entry becomes known.

Observe and update: Following each joint action do as follows: Let a be the action you performed in G_i and let a' be the adversary's action.

- If a, a' are performed for the first time in G_i, update the reward associated with (a, a') in G_i, as observed.
- Update the set of states reached by playing (a, a') in G_i.
- If at this point your record of states reached from this entry contains $K_1 = \max((\lceil \frac{2NTR_{max}}{\epsilon})^3], \lceil -8ln^3(\frac{\delta}{6Nk^2})\rceil) + 1$ elements, mark this entry as *known*, and update the transition probabilities for this entry according to the observed frequencies.

[3] The value 0 given to the adversary does not play an important role here.

As can be seen, R-MAX is quite simple. It starts with an initial estimate for the model parameters that assumes all states and all joint actions yield maximal reward and lead with probability 1 to the fictitious stage-game G_0. Based on the current model, the optimal policy is computed and followed. Following each joint action the agent arrives at a new stage-game, and this transition is recorded in the appropriate place. Once we have enough information about where some joint action leads to from some stage-game, we update the entries associated with this stage-game and this joint action in our model. After each model update, we recompute the policy and repeat the above steps.

4 Optimality and Convergence

In this section we provide the tools that ultimately lead to the proof of the following theorem:

Theorem 1 *Let M be an SG with N states and k actions. Let $0 < \delta < 1$, and $\epsilon > 0$ be constants. Denote the policies for M whose ϵ-return mixing time is T by $\Pi_M(\epsilon, T)$, and denote the optimal expected return achievable by such policies by $Opt(\Pi_M(\epsilon, T))$. Then, with probability of no less than $1 - \delta$ the* R-MAX *algorithm will attain an expected return of $Opt_M(\Pi(\epsilon, T)) - 2\epsilon$ within a number of steps polynomial in $N, k, T, \frac{1}{\epsilon}$, and $\frac{1}{\delta}$.*

In the main lemma required for proving this theorem we show the following: if the agent follows a policy that is optimal with respect to the model it maintains for T steps, it will either attain near-optimal average reward, as desired, or it will update its statistics for one of the unknown slots with sufficiently high probability. This is the *implicit* explore or exploit property of R-MAX: The agent does not know ahead of time whether it is exploring or exploiting – this depends in a large part on the adversary's behavior which it cannot control or predict. However, it knows that it does one or the other, no matter what the adversary does. Using this result we can proceed as follows: As we show, the number of samples required to mark a slot as known is polynomial in the problem parameters, and so is the total number of entries. Therefore, the number of T-step iterations in which non-optimal reward is obtained is bounded by some polynomial function of the input parameters, say T'. This implies that by performing T-step iterations $D = T'R_{max}/\theta$ times, we get that the loss obtained by non-optimal execution (where exploration is performed) is bounded by θ, for any $0 < \theta < 1$.

Before stating our main lemma we extend Kearns and Singh's Simulation Lemma [Kearns and Singh, 1998] to the context of SGs with a slightly improved bound as well.

Definition 2 *Let M and $\bar{M}$ be SGs over the same state and action spaces. We say that $\bar{M}$ is an α-approximation of M if for every state s we have:*

1. *If $P_M(s, t, a, a')$ and $P_{\bar{M}}(s, t, a, a')$ are the probabilities of transition from state s to state t given that the joint action carried out by the agent and the adversary is (a, a'), in M and $\bar{M}$ respectively, then, $P_M(s, t, a, a') - \alpha \leq P_{\bar{M}}(s, t, a, a') \leq P_M(s, t, a, a') + \alpha$*
2. *For every state s, the same stage-game is associated with s in $\bar{M}$ and in M.*

Lemma 1 *Let M and $\bar{M}$ be SGs over N states, where $\bar{M}$ is an $\frac{\epsilon}{NTR_{max}}$-approximation of M, then for every state s, agent policy π, and adversary policy ρ, we have that*

$$|U_{\bar{M}}(s, \pi, \rho, T) - U_M(s, \pi, \rho, T)| \leq \epsilon.$$

Next, we define the notion of an *induced SG*. The definition is similar to the definition of an induced MDP given in [Kearns and Singh, 1998] except for the use of R-MAX. The induced SG is the model used by the agent to determine its policy.

Definition 3 *Let M be an SG. Let L be the set of entries (G_i, a, a') marked* unknown. *That is, if $(G_i, a, a') \in L$ then the entry corresponding to the joint action (a, a') in the stage-game G_i is marked as* unknown. *Define M_L to be the following SG: M_L is identical to M, except that M_L contains an additional state G_0. Transitions and rewards associated with all entries in M_L which are* not *in L are identical to those in M. For any entry in L or in G_0, the transitions are with probability 1 to G_0, and the reward is R_{max} for the agent and 0 for the adversary.*[4]

Given an SG M with a set L of unknown states, and some state s, let $\pi^*_{M_L,s}$ denote the optimal T-step policy for the induced SG M_L starting at s. When M_L and s are clear from the context, we refer to $\pi^*_{M_L,s}$ as the *R-max policy* (because this would be the policy executed by the R-MAX algorithm at this point).

We now state the implicit explore or exploit lemma:

Lemma 2 *Let M be an SG, let L and M_L be as above. Let ρ be an arbitrary policy for the adversary, let s be some state, and let $0 < \alpha < 1$. Then either (1) $|Opt(\Pi_M(\epsilon, T)) - V^s_{R-max}| < \alpha$, where V^s_{R-max} is the expected T-step average reward of $\pi^*_{M_L,s}$ in M starting from s; or (2) An unknown entry will be played in the course of running $\pi^*_{M_L,s}$ on M for T steps starting at s with a probability of at least $\frac{\alpha}{R_{max}}$.*

In practice, we cannot determine ρ, the adversary's policy, ahead of time. Thus, we do not know whether R-MAX will attain near-optimal reward or whether it will reach an unknown entry with sufficient probability. The crucial point is that it will do one or the other, no matter what the adversary does.

We can now outline the proof of Theorem 1. First, we wish to show that the expected average reward is as stated. We must consider three models: M, the real model, M'_L the actual model used, and M', where M' is an $\epsilon/2NTR_{max}$-approximation of M such that the SG induced by M' and L is M'_L. At each T-step iteration of our algorithm we can apply the Implicit Explore or Exploit Lemma to M' and M'_L for the set L applicable at that stage with $\alpha = \epsilon/2$. Hence, at each step either the current R-max policy leads to an average reward that is $\epsilon/2$ close to optimal with respect to the adversary's behavior and the model M' or it leads to an efficient learning policy with respect to the same model. However, because M' is an $\epsilon/2NTR_{max}$-approximation of M, then if the R-max policy is $\epsilon/2$-close to optimal w.r.t. M', it must be ϵ-close to optimal w.r.t. M. We know that the number of T-step phases in which we are exploring can be bounded polynomially. This follows from the fact that we have a polynomial number of parameters to learn (in N and k) and that the probability that we obtain a new, useful statistic is polynomial in ϵ, T and N. Thus, if we choose a large enough (but still polynomial) number of T-step phases, we shall guarantee that our average reward is as close to optimal as we wish.

The above analysis was done assuming we actually obtain the expected value of each random variable. This cannot be guaranteed with probability 1. Yet, we can ensure that the probability that the algorithm fails to attain the expected value of certain parameters be small enough by sampling it a larger (though still polynomial) number of times. This is based on the well-known Chernoff bound. Using this technique one can show that when the variance of some random variable is bounded, we can ensure that we are sufficiently close to its average with probability $1 - \delta$ by using a sufficiently large sample that is polynomial in $1/\delta$.

To remove the assumption that the ϵ-return mixing time is known, we proceed as in [Kearns and Singh, 1998]. From the proof of the algorithm we deduce some polynomial P in the problem parameters such that if T is the mixing-time, then after $P(T)$ steps we are guaranteed, with probability $1 - \delta$, the desired return. We repeatedly execute the algorithm for all values of $T = 1, 2, 3, \ldots$, each time performing $P(T)$ steps. Suppose that T_0 is the mixing time, then after $\sum_{i=1}^{T_0} P(i) = O(P(T_0)^2)$ steps, we will obtain the desired return. ∎

Notice that the R-MAX algorithm does not have a final halting time and will be applied continuously as long as the agent is functioning in its environment. The only caveat is that at some point our current mixing time candidate T will be exponential in the actual mixing time T_0, at which point each step of the algorithm will require an exponential calculation. However, this will occur only after an exponential number of steps. This is true for the E^3 algorithm too.

Another point worth noting is that the agent may never know the values of some of the slots in the game because of the adversary's choices. Consequently, if π is the optimal policy given full information about the game, the agent may actually converge to a policy π' that differs from π, but which yields the best return given the adversary's actual behavior. This return will be no smaller than the return guaranteed by π. The mixing time of π' will, in general, differ from the mixing time of π. However, we are guaranteed that if T_0 is the ϵ-return mixing time of π, and v is its value, after time polynomial in T_0, the agent's actual return will be at least v (subject to the deviations afforded by the theorem).

5 Conclusion

We described R-MAX, a simple reinforcement learning algorithm with guaranteed polynomial-time convergence to near-optimal average reward in zero-sum stochastic games.

R-MAX is an optimistic model-based algorithm that formally justifies the optimism in the face of uncertainty bias. Its analysis is similar, in many respects, to Kearns and Singh's E^3 algorithm. However, unlike the E^3, the agent does not need to explicitly contemplate whether to explore or to exploit. In fact, the agent may never learn an optimal policy for

[4] Any alternative definition of M_L assigning to arbitrary entries in L the same reward as that associated with their counterparts in M is suitable as well.

the game,[5] or it may play an optimal policy without knowing that it is optimal. The "clever" aspect of the agent's policy is that it "offers" a catch to the adversary: if the adversary plays well, and leads the agent to low payoffs, then the agent will, with sufficient probability, learn something that will allow it to improve its policy. Eventually, without too many "unpleasant" learning phases, the agent will have obtained enough information to generate an optimal policy.

R-MAX can be applied to MDPs, repeated games, and SGs provided that the latter satisfy the condition that the value of a state game can be computed efficiently. In particular, all single-controller stochastic game instances covered in [Brafman and Tennenholtz, 2000] fall into this category. However, R-MAX is much simpler conceptually and easier to implement than the LSG algorithm described there. Moreover, it also attains higher payoff: In LSG the agent must pay an additional multiplicative factor ϕ that does not appear in R-MAX.

Two other SG learning algorithms appeared in the literature. Littman [Littman, 1994] describes a variant of Q-learning, called minimax Q-learning, designed for 2-person zero-sum stochastic games. That paper presents experimental results, asymptotic convergence results are presented in [Littman and Szepesvri, 1996]. Hu and Wellman [Hu and Wellman, 1998] consider a more general framework of multi-agent general-sum games. This framework is more general than the framework treated in this paper which dealt with fixed-sum, two-player games. Hu and Wellman based their algorithm on Q-learning as well. They prove that their algorithm converges to the optimal value (defined, in their case, via the notion of Nash equilibrium). However, convergence is in the limit, i.e., provided that every state and every joint action has been visited infinitely often. Note that an adversary can prevent a learning agent from learning certain aspects of the game indefinitely and that the polynomial time convergence of R-MAX to optimal payoff is guaranteed even if certain states and joint actions have never been encountered.

The class of repeated games is another sub-class of stochastic games that satisfies the conditions of R-MAX. In repeated games, $T = 1$, there are no transition probabilities to learn, and we need not use a fictitious stage-game. Therefore, a much simpler version of R-MAX can be used. The resulting algorithm is much simpler and much more efficient than previous algorithms by Megiddo [Megiddo, 1980] and by Banos [Banos, 1968]. Moreover, for these algorithms, only convergence in the limit is proven.

Acknowledgments: We thank Amos Beimel for his help in improving the error bound in Lemma 1 and the anonymous referees for their useful comments. The first author was partially supported by the Paul Ivanier Center for Robotics and Production Management.

References

[Aumann and Maschler, 1995] R. Aumann and M. Maschler. *Repeated Games with Incomplete Information*. MIT Press, 1995.

[Banos, 1968] A. Banos. On pseudo games. *The Annals of Mathematical Statistics*, 39:1932–1945, 1968.

[Brafman and Tennenholtz, 2000] R. Brafman and M. Tennenholtz. A near-optimal polynomial time algorithm for learning in certain classes of stochastic games. *Artificial Intelligence*, 121(1–2):31–47, 2000.

[Hoffman and Karp, 1966] A.J. Hoffman and R.M. Karp. On Nonterminating Stochastic Games. *Management Science*, 12(5):359–370, 1966.

[Hu and Wellman, 1998] J. Hu and M.P. Wellman. Multi-agent reinforcement learning: Theoretical framework and an algorithms. In *Proc. 15th International Conference on Machine Learning*, 1998.

[Kaelbling *et al.*, 1996] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of AI Research*, 4:237–285, 1996.

[Kaelbling, 1993] L. P. Kaelbling. *Learning in Embedded Systems*. The MIT Press, 1993.

[Kearns and Koller, 1999] M. Kearns and D. Koller. Efficient reinforcement learning in factored mdps. In *Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 740–747, 1999.

[Kearns and Singh, 1998] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. In *Int. Conf. on Machine Learning*, 1998.

[Littman and Szepesvri, 1996] M. L. Littman and Csaba Szepesvri. A generalized reinforcement-learning model: Convergence and apllications. In *Proc. 13th Intl. Conf. on Machine Learning*, pages 310–318, 1996.

[Littman, 1994] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. 11th Intl. Conf. on Machine Learning*, pages 157–163, 1994.

[Megiddo, 1980] N. Megiddo. On repeated games with incomplete information played by non-bayesian players. *International Journal of Game Theory*, 9:157–167, 1980.

[Monderer and Tennenholtz, 1997] D. Monderer and M. Tennenholtz. Dynamic Non-Bayesian Decision-Making. *J. of AI Research*, 7:231–248, 1997.

[Moore and Atkenson, 1993] A. W. Moore and C. G. Atkenson. Prioratized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13, 1993.

[Schmidhuber, 1991] J. H. Schmidhuber. Curious model-building control systems. In *Proc. Intl. Joint Conf. on Neural Networks*, pages 1458–1463, 1991.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[Sutton, 1990] R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proc. of the 7th Intl. Conf. on Machine Learning*. Morgan Kaufmann, 1990.

[Tadepalli and Ok, 1998] P. Tadepalli and D. Ok. Model-based average reward reinforcement learning. *Artificial Intelligence*, 100:177–224, 1998.

[5] An agent need not employ an optimal policy to obtain an optimal reward if the adversary plays sub-optimally.

From Q(λ) to Average Q-learning: Efficient Implementation of an Asymptotic Approximation

Frédérick Garcia* **and Florent Serre**
INRA - Unité de Biométrie et Intelligence Artificielle.
BP 27, Auzeville. 31326 Castanet Tolosan cedex, France.
*fgarcia@toulouse.inra.fr

Abstract

Q(λ) is a reinforcement learning algorithm that combines Q-learning and TD(λ). Online implementations of Q(λ) that use eligibility traces have been shown to speed basic Q-learning. In this paper we present an asymptotic analysis of Watkins' Q(λ) with accumulative eligibility traces. We first introduce an asymptotic approximation of Q(λ) that appears to be a gain matrix variant of basic Q-learning. Using the ODE method, we then determine an optimal gain matrix for Q-learning that maximizes its rate of convergence toward the optimal value function Q^*. The similarity between this optimal gain and the asymptotic gain of Q(λ) explains the relative efficiency of the latter for $\lambda > 0$. Furthermore, by minimizing the difference between these two gains, optimal values for the λ parameter and the decreasing learning rates can be determined. This optimal λ strongly depends on the exploration policy during learning. A robust approximation of these learning parameters leads to the definition of a new efficient algorithm called AQ-learning (Average Q-learning), that shows a close resemblance to Schwartz' R-learning. Our results have been demonstrated through numerical simulations.

1 Introduction

TD(λ) and Q-learning are undoubtedly two of the most important methods that have been proposed in Reinforcement Learning. Q-learning is a simple direct algorithm for learning optimal policies, and the TD(λ) rule is used in many existing reinforcement learning and stochastic dynamic programming algorithms for efficiently evaluating a given policy.

The Q-learning algorithm basically implements the TD(0) update rule. In order to improve the convergence rate of Q-learning, it has been proposed to integrate the TD(λ), $\lambda > 0$, with the Q-learning update rule. That resulted in the family of Q(λ) algorithms [Watkins, 1989; Peng and Williams, 1994]. It is commonly believed that Q(λ), $\lambda > 0$, improves the Q-learning behaviour, but no definite theoretical analysis of Q(λ) has come in support of this conjecture: there is no proof of convergence for Q(λ), and as for TD(λ), the issue of the rational choice of λ remains [Singh and Dayan, 1998; Kearns and Singh, 2000].

The use of Q(λ) is hampered by another important difficulty. Eligibility trace implementations of Q(λ) based on look-up tables require many updates of Q-value and trace components after each transition. The time complexity of this approach may be very high for large state and action spaces. In spite of several recent attempts [Cichosz, 1995; Wiering and Schmidhuber, 1997], the multi-component update rule of Q(λ) remains problematic.

In this paper we propose a new algorithm that eliminate the above-mentioned shortcoming of Q(λ). Our approach relies on an asymptotic approximation of Watkins' Q(λ) (section 2 and 3), that appears to be a gain matrix variant of basic Q-learning. Then an optimal convergence rate analysis of Q-learning based on the ODE method leads us to determine an optimal gain matrix for Q-learning (section 4). The similar structure of these two gains explains the relative efficiency of Q(λ). Furthermore, by minimizing the difference between these two gains, we are able to determine near-optimal λ and learning rates that allow en efficient implementation of a new reinforcement learning algorithm called AQ-learning (average Q-learning), which shows a close resemblance to Schwartz' R-learning [Schwartz, 1993] (section 5). We present in section 6 some experimental results that confirm the soundness of the asymptotic analysis and the near-optimal behaviour of AQ-learning.

2 The Watkins' Q(λ) algorithm

Like the TD(λ) and Q-learning reinforcement learning algorithms, Q(λ) can be described within the framework of Markov Decision Processes (MDP) [Puterman, 1994]. Assume a stationary infinite-horizon MDP model that is defined by a state space S of size n_S and an action space A of size n_A, by a Markovian dynamics on S characterized by the transition probabilities $p(s'|s,a)$ of moving from s to s' after applying the action a, and by the instantaneous reward functions $r(s,a,s') \in \mathbb{R}$ associated to each transition (s,a,s').

A policy is a function $\pi : S \rightarrow A$ that assigns an action $a = \pi(s)$ to any possible state s. Given an initial state s_0, following a policy π defines a random trajectory $(s_0, a_0, r_0) \rightarrow (s_1, a_1, r_1) \rightarrow \ldots$ with $a_i = \pi(s_i)$, $r_i = r(s_i, \pi(s_i), s_{i+1})$, according to the probabilities $p(s_{i+1}|s_i, a_i)$.

The optimization problem associated to a discounted Markov Decision Problem is to find a policy π^* that maximizes for any initial state s_0 the expected sum of the discounted rewards along a trajectory $V^\pi(s_0) = E[\sum_{i=0}^{\infty} \gamma^i r(s_i, \pi(s_i), s_{i+1})]$, where $0 \leq \gamma < 1$ is the discount factor.

The value function V^π of a policy π can be directly learnt from observed trajectories by using the TD(λ) algorithm, without maintaining an estimation of the transition probabilities $p(s'|s, \pi(s))$ [Sutton, 1988]. It is also possible to learn directly an estimation of the optimal value function of the problem with the Q-learning algorithm [Watkins, 1989]. Q-learning regularly updates an estimation Q_n of the optimal Q-value function denoted by Q^*:

$$\forall s, a \quad Q^*(s,a) = \sum_{s' \in S} p(s'|s,a) \left(r(s,a,s') + \gamma V^{\pi^*}(s') \right),$$

which is characterized by the optimality equations: $\forall s, a$

$$Q^*(s,a) = \sum_{s' \in S} p(s'|s,a) \left(r(s,a,s') + \gamma \max_{a' \in A} Q^*(s',a') \right).$$

An optimal policy can then be directly derived from Q^* by $\pi^*(s) = \text{argmax}_{a \in A}\, Q^*(s,a), \forall s$.

The principle of Q(λ) is to combine Q-learning and the temporal difference error used in the TD(λ) method, in order to accelerate the convergence of Q-learning. For the discounted value function, the potentially infinite length of the trajectories leads us to consider the Q(λ) methods based on eligibility traces. After each observed transition (s_n, a_n, s_{n+1}, r_n), these algorithms update the estimated value function Q_n by

$$\forall s, a \quad Q_{n+1}(s,a) \leftarrow Q_n(s,a) + \alpha_n(s,a) z_n(s,a) d_n \quad (1)$$

where $d_n = r_n + \gamma \max_{a' \in A} Q_n(s_{n+1}, a') - Q_n(s_n, a_n)$ is the current temporal difference, $\alpha_n(s,a)$ is the stepsize and $z_n(s,a)$ is the eligibility trace.

Usually, stepsizes decay regularly toward 0, or are very small constants. The dynamics of the eligibility trace is more complex. Like in TD(λ), the value $z_n(s,a)$ is increased for the pair (s_n, a_n), and decays exponentially according to the parameter $\lambda \in [0,1]$ for all pairs $(s,a) \neq (s_n, a_n)$. Following this general principle, several eligibility trace dynamics for Q(λ) have been proposed [Watkins, 1989; Peng and Williams, 1994; Singh and Sutton, 1996]. The simplest one, due to Watkins [Watkins, 1989], is directly derived from the accumulative eligibility trace of TD(λ), and has the supplementary effect of resetting to 0 the whole trace vector when the current action a_n is not a *greedy action*, that is $a_n \neq a_n^* \triangleq \text{argmax}_a\, Q_n(s_n, a)$. The complete definition of this trace is given by:

$$\begin{array}{rcl} z_n(s_n,a_n) & \leftarrow & \left\{ \begin{array}{ll} \gamma\lambda z_{n-1}(s_n,a_n) + 1 & \text{if } a_n = a_n^* \\ 1 & \text{if } a_n \neq a_n^* \end{array} \right. \\ z_n(s,a) & \leftarrow & \left\{ \begin{array}{ll} \gamma\lambda z_{n-1}(s,a) & \text{if } a_n = a_n^* \\ 0 & \text{if } a_n \neq a_n^* \end{array} \right. \end{array} \quad (2)$$

$\forall (s,a) \neq (s_n, a_n)$, for $n \geq 0$, with $\gamma < 1$, $\lambda \in [0,1]$, and $\forall (s,a)\ z_{-1}(s,a) = 0$.

Q(λ) with accumulative eligibility trace is a direct generalization of Q-learning; Q(0) exactly corresponds to Q-learning, where $Q_n(s_n, a_n)$ is the only component updated after each transition. Unlike the convergence of Q-learning that has been proved [Bertsekas and Tsitsiklis, 1996; Borkar and Meyn, 2000], the convergence of Q(λ) and its overcoming of Q-learning have only been shown experimentally [Peng and Williams, 1994; Wiering and Schmidhuber, 1997]. In practice, the main drawback of Q(λ) is relative to the update costs of Q_n and z_n that are proportional to the size $n_S n_A$ of $S \times A$ for $\lambda > 0$. For large state-action space problems, the computation time of Q(λ), $\lambda > 0$, can be very high.

3 Asymptotic approximation of Q(λ)

In this section we intend to define an asymptotic approximation of Watkins' Q(λ) when $n \to \infty$. The following assumptions are made. At each iteration the action a_n is chosen with a semi-uniform exploration policy: a greedy action a_n^* is selected with a probability $0 < \tau < 1$, or any other action in A is chosen randomly. Asymptotically, when Q_n has converged toward Q^*, this process defines a Markov chain (s_n, a_n) on $S \times A$. We assume that this Markov chain is regular (recurrent, aperiodic, irreducible) and therefore has a unique invariant distribution $\mu > 0$.

3.1 Asymptotic accumulating trace

Let us consider a trajectory $(s_0, a_0, r_0) \to (s_1, a_1, r_1) \to \cdots (s_n, a_n, r_n)$ obtained by Q(λ), such that Q_n converges toward Q^*. Generalizing our work developed for TD(λ) [Garcia and Serre, 2000], the average asymptotic behaviour of the accumulative trace $z_n(s,a)$ on this trajectory can be calculated:

Proposition 1

$$\forall s \in S,\ a \in A \quad \lim_{n \to \infty} E[z_n(s,a)] = \frac{\mu(s,a)}{1 - \gamma\lambda\tau}.$$

For $\tau \to 1$ the formula is similar to the one developed for TD(λ). For $\tau \to 0$, the trace is regularly set to 0 or 1, and its average value is equal to the invariant distribution μ. In the general case, we can see that the role of the trace $z_n(s,a)$ in the update rule (1) is to increase credit of state-action pairs that occur more frequently, particularly when the factor $1 - \gamma\lambda\tau$ is small.

Proposition 1 gives the average asymptotic value of $z_n(s,a)$ when $n \to \infty$. From the update rule (2), we can derive an approximation of the asymptotic expectation of the trace at time n conditioned on the current state-action pair (s_n, a_n):

$$\bar{z}_n(s,a) = \left\{ \begin{array}{ll} \chi\mu(s,a) + 1 & \text{if } (s,a) = (s_n,a_n),\ a_n = \pi^*(s_n) \\ \chi\mu(s,a) & \text{if } (s,a) \neq (s_n,a_n),\ a_n = \pi^*(s_n) \\ 1 & \text{if } (s,a) = (s_n,a_n),\ a_n \neq \pi^*(s_n) \\ 0 & \text{if } (s,a) \neq (s_n,a_n),\ a_n \neq \pi^*(s_n) \end{array} \right.$$

$\forall s, a$, with $\chi = \frac{\gamma\lambda}{1 - \gamma\lambda\tau}$.

3.2 Asymptotic Q(λ)

We consider the asymptotic approximation AQ(λ) of Q(λ) obtained by substituting in the Q(λ) update rule (1) the trace factor $z_n(s,a)$ by its approximation $\bar{z}_n(s,a)$.

Let us denote by A_n the $n_S n_A \times n_S n_A$ diagonal matrix with diagonal entries $\alpha_n(s,a)$, and by Γ^z the $n_S n_A \times n_S n_A$ *eligibility matrix*

$$\Gamma^z = I + \chi \begin{pmatrix} \overset{n_S}{\overbrace{\begin{matrix} | & & | \\ \mu & \dots & \mu \\ | & & | \end{matrix}}} & \overset{n_S(n_A-1)}{\overbrace{\begin{matrix} \\ 0 \\ \\ \end{matrix}}} \end{pmatrix}$$

where the state-action pairs of $S \times A$ are ordered as follows:$(s_1, \pi^*(s_1)), \dots, (s_{n_S}, \pi^*(s_{n_S}))$, $\dots, (s_i, a_j | a_j \neq \pi^*(s_i)), \dots$ (the column headings indicate the block dimensions).

Let $D_n \in I\!R^{n_S n_A}$ be the temporal difference vector :

$$D_n^T = (0, \dots, 0, D_{n(s_n,a_n)} = d_n, 0, \dots, 0)$$

In vector notation, the asymptotic approximation AQ(λ) of Q(λ) is defined by

$$Q_{n+1} \leftarrow Q_n + A_n \Gamma^z D_n. \tag{3}$$

4 Optimal gain matrix for Q-learning

One can see that (3) is a gain matrix variant of the basic Q-learning algorithm:

$$Q_{n+1} \leftarrow Q_n + \frac{1}{n} D_n.$$

Using a gain matrix in order to guide and accelerate the convergence of a stochastic adaptive algorithm is a well-known approach in stochastic approximation theory [Benveniste *et al.*, 1990; Kushner and Yin, 1997]. In this section, we calculate the optimal gain matrix for Q-learning that maximizes its rate of convergence.

The ordinary differential equation (ODE) method provides some analytic tools for analysing the convergence of the general stochastic algorithm

$$\theta_{n+1} \leftarrow \theta_n + \frac{1}{n} H(\theta_n, X_n)$$

where θ_n is the current estimation of the target θ^*, and X_n is the observed input random vector at time n. Classic reinforcement learning algorithms like Q-learning or TD(λ) have already been analysed with the ODE method [Bertsekas and Tsitsiklis, 1996; Borkar and Meyn, 2000], in order to prove their convergence.

Another potential application of the ODE method concerns the optimization of the rate of convergence of stochastic algorithms. One can show that under some general stability conditions [Benveniste *et al.*, 1990] the new algorithm

$$\theta_{n+1} \leftarrow \theta_n + \frac{1}{n} \Gamma^* H(\theta_n, X_n)$$

where Γ^* is the gain matrix defined by

$$\Gamma^* = -h_\theta(\theta^*)^{-1},$$

and h_θ is the Jacobian matrix of the function

$$h(\theta) = \lim_{n \to \infty} E_\theta[H(\theta, X_n)]$$

still converges toward θ^*, with a minimal asymptotic variance $\lim_{n\to\infty} nE[\| \theta_n - \theta^* \|^2]$.

This approach, that has already been applied to Q-learning in finite horizon [Garcia and Ndiaye, 1998] and TD(λ) [Garcia and Serre, 2000], is developed here for Q-learning. We have $\theta_n = Q_n \in I\!R^{n_S n_A}$, $\theta^* = Q^*$, $X_n = (s_n, a_n, s_{n+1})$ and $H(\theta_n, X_n) = D_n$. The Jacobian matrix and its inverse can be calculated:

$$h_Q(Q^*) = \begin{pmatrix} \overset{n_S}{\overbrace{M^*(\gamma P^* - I)}} & \overset{n_S(n_A-1)}{\overbrace{\begin{matrix} 0 \\ -\overline{M} \end{matrix}}} \\ \overline{M}\gamma\overline{P} & \end{pmatrix},$$

$$\Gamma^* = \begin{pmatrix} \overset{n_S}{\overbrace{(I - \gamma P^*)^{-1} M^{*-1}}} & \overset{n_S(n_A-1)}{\overbrace{\begin{matrix} 0 \\ \overline{M}^{-1} \end{matrix}}} \\ \gamma\overline{P}(I - \gamma P^*)^{-1} M^{*-1} & \end{pmatrix},$$

where M^* is the $n_S \times n_S$ diagonal matrix with diagonal coefficients $\mu(s, \pi^*(s))$, $\overline{M}$ is the $n_S(n_A-1) \times n_S(n_A-1)$ diagonal matrix with diagonal coefficients $\mu(s,a)$ for $a \neq \pi^*(s)$, P^* is the $n_S \times n_S$ transition matrix of the Markov chain on S defined by π^*, and $\overline{P}$ the $n_S(n_A-1) \times n_S$ rectangular matrix of coefficients $\overline{P}_{(s,a),s'} = p(s'|s,a)$ for $a \neq \pi^*(s)$.

By developing this expression, we show

Proposition 2
The optimal gain matrix for Q-learning is

$$\Gamma^* = M^{-1} + \gamma \begin{pmatrix} \overset{n_S}{\overbrace{P_\gamma^* M^{*-1}}} & \overset{n_S(n_A-1)}{\overbrace{0}} \end{pmatrix}, \tag{4}$$

where

$$M = \begin{pmatrix} M^* & 0 \\ 0 & \overline{M} \end{pmatrix}$$

is the diagonal matrix with diagonal coefficients $\mu(s,a)$, and P_γ^* is the $n_S(n_A-1) \times n_S$ rectangular matrix of coefficients

$$P^*_{\gamma(s,a),s'} = \sum_{n=0}^{\infty} \gamma^n p_n^*(s' \mid s, a),$$

where $p_n^*(s' \mid s, a)$ is the probability of moving from s to s' by applying the first action a and then by following the optimal policy π^* on the next n transitions.

This optimal gain Γ^* leads to an "ideal" algorithm, called Optimal-Q-learning:

$$Q_{n+1} \leftarrow Q_n + \frac{1}{n} \Gamma^* D_n. \tag{5}$$

This algorithm theoretically maximizes the rate of convergence of Q-learning, by minimizing its asymptotic variance $\lim_{n\to\infty} nE[\| Q_n - Q^* \|^2]$. One can see that the structure of Γ^* implies that when a greedy action is chosen in s_n, all the $Q_n(s,a)$ components have to be updated, and when an exploratory action has been selected, only $Q_n(s_n, a_n)$ is modified. Note that Γ^* is independent of the reward functions $r()$, and depends on the parameter τ that characterizes the exploration policy, through the μ invariant distribution.

5 Average Q-learning

Optimal-Q-learning cannot be directly implemented since it would require the costly online estimation of the P^*_γ coefficients. However, as shown next, it is possible to determine some λ and α_n parameters that minimize the distance between the gain matrix of AQ(λ) and Γ^*, and that allow an efficient implementation of the corresponding AQ(λ) algorithm.

5.1 Optimization of AQ(λ)

To be consistent with the definition of Γ^* that assumes a stepsize equal to $\frac{1}{n}$, we consider the case $A_n = \frac{1}{n}A$, where

$$A = \begin{array}{c} \begin{matrix} n_S & n_S(n_A-1) \end{matrix} \\ \begin{pmatrix} A^* & 0 \\ 0 & \overline{A} \end{pmatrix} \end{array}$$

is the $n_S n_A \times n_S n_A$ diagonal matrix with constant coefficients $\alpha(s,a)$. We propose to optimize the choice of the α and λ parameters by minimizing the quadratic norm $\| A\Gamma^z - \Gamma^* \|$ of the difference between the gain of AQ(λ) and the optimal gain of Q-learning.

We have

$$A\Gamma^z = \begin{pmatrix} A^*(I + \chi M^* U^*) & 0 \\ \chi \overline{A} M U & \overline{A} \end{pmatrix},$$

where the $n_S \times n_S$ matrix U^* and the $n_S(n_A-1) \times n_S$ matrix $\overline{U}$ have coefficients equal to 1.

Making equal the lower right blocks leads to the choice $\overline{A} = \overline{M}^{-1}$. A numerical analysis of the upper left blocks, similar to the one developed in [Garcia and Serre, 2000] for TD(λ), results in the choice $A^* = M^{*-1}$, and thus $A = M^{-1}$. Then

$$A\Gamma^z = M^{-1} + \chi \begin{array}{c} \begin{matrix} n_S & & n_S(n_A-1) \end{matrix} \\ \begin{pmatrix} | & & | & \\ 1 & \dots & 1 & 0 \\ | & & | & \end{pmatrix} \end{array}.$$

This equality exhibits some similarity with equation (4). As we can note, the structures of the two gains are identical, which might explain the efficiency of Q(λ) for $\lambda > 0$.

Minimizing

$$f(\chi) = \sum_{s,a,s'} \left(\chi - \gamma \frac{P^*_{\gamma(s,a),s'}}{\mu(s', \pi^*(s'))} \right)^2$$

yields

$$\chi_{opt} = \gamma \frac{1}{{n_S}^2 n_A} \sum_{s,a,s'} \frac{P^*_{\gamma(s,a),s'}}{\mu(s', \pi^*(s'))}.$$

Like for Γ^*, this χ_{opt} value cannot be exactly implemented in practice in AQ(λ). However, $\chi_{opt} \approx \frac{1}{\tau}\frac{\gamma}{1-\gamma}$ for uniform MDPs where all the transition probabilities are nearly equal to $\frac{1}{n_S}$ (then $P^*_\gamma(s' \mid s,a) \approx \frac{1}{n_S}\frac{1}{1-\gamma}$, $\mu(s', \pi^*(s')) \approx \frac{\tau}{n_S}$ and so $\chi_{opt} \approx \frac{1}{\tau}\frac{\gamma}{1-\gamma}$). It appears experimentally that for large random MDP, this approximation is still valid.

The value $\chi_{opt} \approx \frac{1}{\tau}\frac{\gamma}{1-\gamma}$ leads to $\lambda_{opt} = \frac{1}{\gamma}\frac{\chi_{opt}}{1+\tau\chi_{opt}} \approx \frac{1}{\tau}$. This value greater than 1 could be surprising. In fact convergence of eligibility trace methods requires that $\gamma\lambda < 1$ [Bertsekas and Tsitsiklis, 1996], and so $\lambda = \frac{1}{\tau}$ could be implemented in Q(λ) assuming $\tau > \gamma$. In case $\tau \leq \gamma$, $\frac{1}{\tau}$ is no more an admissible value for λ, and the distance is minimized for $\lambda = \frac{1}{\gamma}$ on the boundary.

We are thus able to define a new optimized update rule from the asymptotic approximation (3) where $\chi = \frac{1}{\tau}\frac{\gamma}{1-\gamma}$ and $\forall n, s, a$, $\alpha_n(s,a) = \frac{1}{n}\frac{1}{\mu(s,a)}$:

$$Q_{n+1} \leftarrow Q_n + \frac{1}{n}\Gamma D_n, \tag{6}$$

where $\Gamma = M^{-1} + \frac{1}{\tau}\frac{\gamma}{1-\gamma} \begin{array}{c} \begin{matrix} & n_S & & n_S(n_A-1) \end{matrix} \\ \begin{pmatrix} | & & | & \\ 1 & \dots & 1 & 0 \\ | & & | & \end{pmatrix} \end{array}$.

5.2 Efficient implementation of AQ-learning

A direct implementation of the update rule (6) would have a complexity similar to Q(λ), proportional to $n_S \times n_A$. Fortunately, an equivalent expression can be derived. It only requires a small number of updates per iteration, independent of the number of states and actions.

Proposition 3
The update rule (6) is equivalent to

$$\begin{cases} W_{n+1}(s_n, a_n) \leftarrow W_n(s_n, a_n) + \frac{1}{n}\frac{1}{\mu(s_n, a_n)} d_n, \\ \rho_{n+1} \leftarrow \rho_n + \frac{\gamma}{\tau}\frac{1}{n} d_n \quad \textit{if } a_n = \pi^*(s_n), \end{cases}$$

$$d_n = r_n - \rho_n + \gamma \max_a W_n(s_{n+1}, a) - W_n(s_n, a_n),$$

$$\textit{and} \quad \forall s, a \quad Q_n(s,a) = W_n(s,a) + \frac{1}{1-\gamma}\rho_n.$$

Proof: From (6) we have $Q_n = \sum_{p<n} \frac{1}{p} M^{-1} D_p + \frac{1}{\tau}\frac{\gamma}{1-\gamma}\sum_{p<n}\frac{1}{p} d_p e \times \mathbb{1}_{a_p = \pi^*(s_p)}$, where e denotes the vector with coefficients equal to 1, and $\mathbb{1}_{x=y}$ equals 0 or 1 according as $x \neq y$ or $x = y$. We then define the new relative value function W_n and the scaling factor ρ_n by

$$\begin{cases} W_n = \sum_{p<n} \frac{1}{p} M^{-1} D_p, \\ \rho_n = \sum_{p<n} \frac{\gamma}{\tau}\frac{1}{p} d_p \times \mathbb{1}_{a_p = \pi^*(s_p)}. \end{cases}$$

The d_n error can be expressed as a function of ρ_n and W_n:

$$\begin{aligned} d_n &= r_n + \gamma \max_a Q_n(s_{n+1}, a) - Q_n(s_n, a_n) \\ &= r_n - \rho_n + \gamma \max_a W_n(s_{n+1}, a) - W_n(s_n, a_n). \quad \square \end{aligned}$$

Note that the greedy action a^*_n is defined by $a^*_n = \operatorname{argmax}_a Q_n(s_n, a) = \operatorname{argmax}_a W_n(s_n, a)$. The update

rule of W_n can be implemented by replacing the stepsize $\frac{1}{\mu(s_n,a_n)n}$ with its approximation $\frac{1}{N_n(s_n)p(a_n)}$, where $N_n(s_n)$ is the number of times the state s_n has been visited at time n, and $p(a_n)$ equals τ or $\frac{1-\tau}{n_A-1}$ whether $a_n = a_n^*$ or $a_n \neq a_n^*$. Similarly, the condition $a_n = \pi^*(s_n)$ for updating ρ_n might be replaced by $a_n = a_n^*$. This leads to the following algorithm AQ-learning, or Average Q-learning:

AQ-learning Algorithm

$W = \rho = N = 0$;
Choose s_0 in S;
for $n \leftarrow 0$ **until** $N_{tot} - 1$ **do**
  Choose a_n in A (τ-exploration policy);
  $N(s_n) \leftarrow N(s_n) + 1$;
  $(s_{n+1}, r_n) \leftarrow$ Simulate (s_n, a_n);
  $d \leftarrow r_n - \rho + \gamma \max_a W(s_{n+1}, a) - W(s_n, a_n)$;
  $W(s_n, a_n) \leftarrow W(s_n, a_n) + \frac{1}{N(s_n)p(a_n)} d$;
  if $a_n = a_n^*$ **then**
    $\rho \leftarrow \rho + \frac{\gamma}{\tau}\frac{1}{n} d$;
for $s \in S, a \in A$ **do**
  $Q(s,a) \leftarrow W(s,a) + \frac{1}{1-\gamma}\rho$;
return Q, W, ρ;

In that algorithm, N_{tot} is the length of the learning trajectory. AQ-learning is particularly interesting, since it only requires 3 updates per iteration, namely $W_n(s_n, a_n)$, ρ_n and $N_n(s_n, a_n)$. AQ-learning should therefore improve the performance of Watkins' Q(λ) for two distinct reasons. First, AQ-learning is an optimized approximation of Q(λ) that minimizes the asymptotic variance around Q^*. Second, like Q-learning, AQ-learning is an asynchronous algorithm that only updates one component of the value function (here W_n) per iteration. It is clear that for large state-action space problems, this feature is of crucial importance regarding the time complexity of the algorithm, as it is shown in simulations in the next section.

Amazingly it appears that AQ-learning is very close to the R-learning algorithm proposed by Schwartz [Schwartz, 1993]. R-learning is supposed to converge to gain optimal policies, that maximize the average-cost

$$\rho^\pi = \lim_{N\to\infty} E\left[\frac{1}{N}\sum_{i=0}^{N-1} r(s_i, \pi(s_i), s_{i+1})\right].$$

AQ-learning and R-learning are identical when $\gamma \to 1$, with a correct choice of the R-learning learning rates. The fact that AQ-learning is a near-optimal gain matrix variant of Q-learning could explain why several experiments have demonstrated that R-learning could outperform Q-learning, despite the fact that these two algorithms have not been designed for the same optimality criterion.

We do not provide in this article a convergence proof for AQ-learning. We think, however, that it might be possible to establish such a proof either by checking the stability condition of the gain matrix $M^{-1}\Gamma^z$ for $\chi = \frac{1}{\tau}\frac{\gamma}{1-\gamma}$, or by exploiting the ODE approach of Borkar [Borkar, 1996] for two-time scale algorithms (it might be necessary in that case to modify the ρ_n update rule).

6 Simulations

We made numerical experiments in order to study the relative performance of Q-learning, Optimal-Q-learning, Watkins' Q(λ) and AQ-learning. These reinforcement learning algorithms were applied to randomly generated Markov Decision Problems, with all components $p(s'|s,a) > 0$, and with state rewards $r(s) \in [0,1]$.

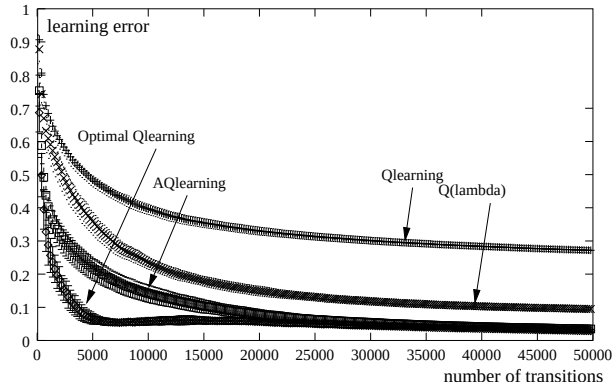


Figure 1: Learning error $\frac{\|V_n - V^*\|}{\|V^*\|}$

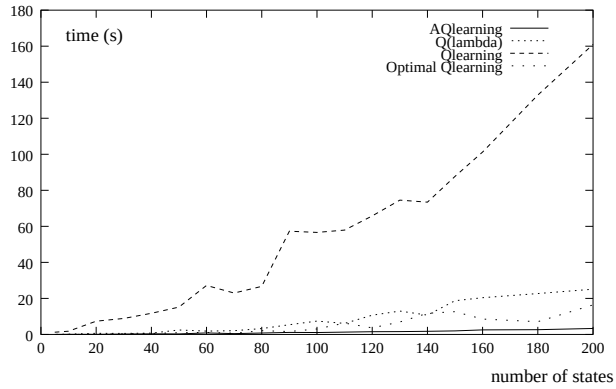


Figure 2: Computation time for $\frac{\|V_n - V^*\|}{\|V^*\|} < 0.05$

We present here two kinds of learning results that fairly describe the general behaviour of these algorithms. For both of them, the value functions Q_n, W_n and ρ_n were updated on a unique long trajectory on $S \times A$ with random initial state. All parameters were initially set to 0. The stepsizes $\alpha_n(s,a)$ were equal to $\frac{1}{N_n(s,a)}$. We chose a semi-uniform exploration strategy with $\tau > \gamma$. For each problem, V^*, Γ^* and λ_{opt} were exactly calculated from the data $p()$ and $r()$.

Figure 1 shows the relative learning error $\frac{\|V_n - V^*\|}{\|V^*\|}$ as a function of n (average value and standard deviation on 20

random trajectories). The size of the MDP was $n_S = 100$, $n_A = 5$, with $\gamma = 0.8$ and $\tau = 0.9$. This graph illustrates the different rates of convergence of Q-learning, AQ-learning, Q(λ) for $\lambda = \lambda_{opt}$, and Optimal-Q-learning. These experimental results confirm that the faster rate of convergence is obtained with Optimal-Q-learning, and that AQ-learning converges nearly as fast as this ideal algorithm. AQ-learning even performs slightly better than Q(λ) with the same learning rates and the λ_{opt} value. These three algorithms appear to be considerably faster than Q-learning.

Figure 2 illustrates for the same algorithms the different computation times required to achieve a relative learning error equal to 5%, as a function of the size n_S of the state space, for a constant number of actions $n_A = 5$, with $\gamma = 0.7$ and $\tau = 0.9$. We first observe on that graph that Q-learning is very slow, despite the fact that it only updates one component per iteration. But the main observation is that AQ-learning can be really more efficient than Q(λ), especially when the number of states is important; for 200 states, AQ-learning is 10 times faster than Q(λ).

7 Conclusions

We have shown that an asymptotic approximation of Q(λ) could be efficiently implemented through two simple update rules on a relative value function $W_n(s,a)$ and on a scaling factor ρ_n. This new algorithm AQ-learning can be described as an average variant of Q-learning for the discounted criterion. Its update complexity is independent of the number of states and actions, and experimental results show that its convergence toward the optimal value function is always faster than for Q-learning or Q(λ).

Our results have been obtained for the Watkins' Q(λ) with accumulating eligibility trace. It would be interesting to apply the same approach to the case of replacing traces [Singh and Sutton, 1996], or to Peng and Williams' Q(λ) [Peng and Williams, 1994] where the trace is never reset to 0, and for which an efficient implementation has been proposed by Wiering and Schmidhuber [Wiering and Schmidhuber, 1997].

Another direction for future work concerns the development of new reinforcement learning algorithms that approximate Optimal-Q-learning. Furthermore, we have seen that the Γ^* optimal gain for Q-learning depends on the exploration policy parameterized by τ. The question of finding the best value of τ remains open.

Finally, we need a better understanding of the relation between AQ-learning the Schwartz' R-learning algorithm [Schwartz, 1993].

References

[Benveniste *et al.*, 1990] A. Benveniste, M. Metivier, and P. Priouret. *Adaptive Algorithms and Stochastic Approximation*. Springer-Verlag, Berlin,New York, 1990.

[Bertsekas and Tsitsiklis, 1996] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont (MA), 1996.

[Borkar and Meyn, 2000] V. S. Borkar and S. P. Meyn. The O.D.E. method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal of Control and Optimization*, 38:447–469, 2000.

[Borkar, 1996] V. S. Borkar. Stochastic approximation with two-time scales. *System and Control Letters*, 29:291–294, 1996.

[Cichosz, 1995] P. Cichosz. Truncating temporal differences: On the efficient implementation of TD(λ) for reinforcement learning. *Journal of Artificial Intelligence Research (JAIR)*, 2:287–318, 1995.

[Garcia and Ndiaye, 1998] F. Garcia and S. Ndiaye. A Learning Rate Analysis of Reinforcement-Learning Algorithms in Finite-Horizon. In *International Conference on Machine Learning*, volume 15, Madison, USA, 1998.

[Garcia and Serre, 2000] F. Garcia and F. Serre. Efficient asymptotic approximation in temporal difference learning. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000)*, Berlin, 2000.

[Kearns and Singh, 2000] M. Kearns and S.P. Singh. "bias-Variance" error bounds for temporal difference updates. In *Proceedings of the COLT 2000 conference*, 2000.

[Kushner and Yin, 1997] H. J. Kushner and G. G Yin. *Stochastic Approximation Algorithms and Applications*. Springer, 1997.

[Peng and Williams, 1994] J. Peng and R. J. Williams. Incremental Multi-Step Q-Learning. In *International Conference on Machine Learning*, volume 11, pages 226–232, 1994.

[Puterman, 1994] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. Wiley-Interscience, New York, 1994.

[Schwartz, 1993] A. Schwartz. A Reinforcement Learning Method for Maximizing Undiscounted Rewards. In *International Conference on Machine Learning*, volume 10, 1993.

[Singh and Dayan, 1998] S. P. Singh and P. Dayan. Analytical Mean Squared Error Curves for Temporal Difference learning. *Machine Learning*, 1998.

[Singh and Sutton, 1996] S. P. Singh and R. S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 1996.

[Sutton, 1988] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[Watkins, 1989] C. J. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989.

[Wiering and Schmidhuber, 1997] M. A. Wiering and J. Schmidhuber. Fast online Q(λ). *Machine Learning*, 1997.

Exploiting Multiple Secondary Reinforcers in Policy Gradient Reinforcement Learning

Greg Grudic
IRCS and GRASP Lab
University of Pennsylvania
Philadelphia, PA, USA
grudic@linc.cis.upenn.edu

Lyle Ungar
Computer and Information Science
University of Pennsylvania
Philadelphia, PA, USA
ungar@cis.upenn.edu

Abstract

Most formulations of Reinforcement Learning depend on a single reinforcement reward value to guide the search for the optimal policy solution. If observation of this reward is rare or expensive, converging to a solution can be impractically slow. One way to exploit additional domain knowledge is to use more readily available, but related quantities as **secondary reinforcers** to guide the search through the space of all policies. We propose a method to augment Policy Gradient Reinforcement Learning algorithms by using prior domain knowledge to estimate desired relative levels of a set of secondary reinforcement quantities. RL can then be applied to determine a policy which will establish these levels. The primary reinforcement reward is then sampled to calculate a gradient for each secondary reinforcer, in the direction of increased primary reward. These gradients are used to improve the estimate of relative secondary values, and the process iterates until reward is maximized. We prove that the algorithm converges to a local optimum in secondary reward space, and that the rate of convergence of the performance gradient estimate in secondary reward space is independent of the size of the state space. Experimental results demonstrate that the algorithm can converge many orders of magnitude faster than standard policy gradient formulations.

1 Introduction

Reinforcement Learning (RL) is a trial and error procedure by which an agent learns to improve the reward it receives from the environment. At the core of all RL algorithms is a search procedure that directs the hypothesis and test process by which the agent discovers how to improve its behaviour. The efficiency of an RL algorithm is therefore governed by the size of the agent's search space (i.e. problem domain size) and the effectiveness of the search procedure used. When the problem domain is small enough, or enough prior knowledge is used to direct search, RL algorithms can yield effective solutions which are not easily achievable by other means [Kaelbling *et al.*, 1996].

There is little doubt that mammals (as well as other biological systems) successfully use some form of reinforcement learning [Sutton and Barto, 1990]. Furthermore, the environments where most mammals live are varied, complex, and unpredictable, implying a large problem domain that cannot be blindly searched to the animals advantage. Thus, mammals must employ intelligent search strategies in order to learn how to improve the rewards they receive. One search strategy employed by mammals involves the use of *secondary reinforcers*, which are stimuli that can be associated with the primary reinforcement reward, and thus have similar reinforcing characteristics. Secondary reinforcers are useful to an organism if it can more easily learn how to obtain the stimuli associated with secondary reinforcement than those associated with a primary reinforcement. In general, organisms seek the goals of successful breeding and long term survival. These however can rarely be sampled and therefore provide little guidance in the search for a survival 'policy'. More immediate stimuli such as fear, hunger, cold, pain or pleasure can be much more readily observed and have direct consequences for the animal's long term reward. For example, if a mouse learns that hanging around a kitchen (a secondary reinforcer) is sometimes associated with the primary reinforcer of assuaging hunger (and not starving), then, because the kitchen is always there while the food is not, the mouse can more easily learn to modify its behavior to attain the secondary reinforcer than the primary one.

Minsky [Minsky, 1963] was one of the first to argue that the use of secondary reinforcers may be an important ingredient to creating an artificial intelligence that is able to learn through reinforcement learning. Indeed, one of the motivating factors behind the widely successful temporal-difference reinforcement learning algorithm (TD(λ)) is the notion of secondary reinforcers [Sutton and Barto, 1998].

In this work we propose to extend the use of secondary reinforcers to improve the rate of convergence of RL algorithms. To date, the use of secondary reinforcers in RL has largely been limited to simply noting which stimuli are correlated with the primary reward and having the agent learn to maximize these stimuli independently. However, this simple notion of blindly maximizing all secondary reinforcer stimuli is naive because there are potentially complicated interactions among secondary reinforcers associated with a problem domain. In fact, the agent is likely to gain an advantage by con-

trolling the *relative* amounts of stimuli from each reinforcer, rather than concentrating on achieving maximal values. Even in the simple example of a mouse learning to spend time in the kitchen because food may fall to the floor, the mouse may not be best served by spending all of his time in the kitchen because a cat may also at times wander in, and greatly curtail the mouse's long term survival. Thus the mouse might want to control the percentage of time it spends in kitchen in order to minimize the risk of running into the cat. Hence, even in simple examples, controlling how much of each secondary reinforcer the agent should experience is important.

Consider the example of a student who must write a set of 4 hour exams (one per day), with 20 hours between when one exam finishes and the next exam is scheduled to begin. She knows that both studying and getting a good night's sleep are associated with getting good marks on an exam, so there are two secondary reinforcers which she can use. She can modify her behavior in such a way that she will study (the first secondary reinforcer) for a longer or shorter period, or conversely she will learn how to get a longer or shorter night's sleep (the other secondary reinforcer). However, if she simply spends the next 20 hours sleeping, or if she forces herself to study for the next 20 hours, she will do poorly on the exam. Thus, in order to maximize her exam marks, she will need to determine what policy actions (drink more coffee, spend more time at the library, drink more warm milk etc) she must perform in order to get the right ratio of the two secondary reinforcers (i.e. sleep and study).

1.1 SRPG Method Overview

In this work we propose a new algorithm called *Secondary Reinforcers Policy Gradient* (SRPG) for solving the problem of determining the relative amounts of secondary reinforcers an agent needs to experience in order to increase its primary reward. Our framework assumes that the agent starts with prior domain knowledge defining a set of secondary reinforcers and in what relative amounts these should be experienced. We further assume that the secondary reinforcers are more readily observed than the primary reward, and thus the agent can more easily learn strategies which allow it to experience these stimuli in the appropriate ratios.

Our algorithm consists of the following repeated steps:

STEP I: Use prior domain knowledge to establish the initial estimate of desired relative amounts of secondary reinforcer stimuli. Use RL to learn a policy which will yield these levels of stimuli. During this step, the agent need never experience the primary stimulus, only the secondary ones.

STEP II: Based on the policy established in STEP I a gradient on the primary reward is calculated with respect to each secondary reinforcer stimulus. These gradients are then used to update the desired relative amounts of the secondary reinforcers and STEP I of the algorithm is repeated.

If the important secondary reinforcers and the initial guess at their relative ratios is chosen wisely, then we show both theoretically and experimentally that the agent can improve the reward it receives significantly faster than if no such prior knowledge is used. Because in RL problems it is often the case that primary reinforcers are infrequently experienced by the agent, the rate of convergence measure used in this work is the number of times the agent needs to experience a primary reward to achieve a specific amount of reward.

In applying our proposed framework to the exam studying example given above, the student may start by knowing that sleep and study are both important stimuli, and that she will likely need eight hours of sleep and twelve hours of study. Thus the student can spend many 20 hour periods learning which actions allow her to sleep for eight hours and study for twelve, without ever taking an exam. But by the time she does take an exam, she is quite good at sleeping for eight hours and studying for twelve. Once she observes her first exam grades, she can learn to maximize her grade by systematically adjusting her sleep/wake ratio (i.e. her secondary reinforcers) which prior knowledge tells her are directly related to her mark, rather than just adjusting actions such as drinking milk or coffee based on a primary reward (i.e. her exam mark) which she only rarely observes.

The proposed algorithm is based on methods derived from Policy Gradient Reinforcement Learning (PGRL) [Williams, 1987; 1992; Baird and Moore, 1999; Sutton *et al.*, 2000; Konda and Tsitsiklis, 2000; Baxter and Bartlett, 2000; Grudic and Ungar, 2000b; 2000a], rather than Value Function Reinforcement Learning (VFRL) methods such as Q learning [Sutton and Barto, 1998]. The motivation for this is threefold. 1) VFRL methods work by learning a state-action value function which defines the value of executing each action in each state, thus allowing the agent to choose the most valuable action in each state. As a result, the search space grows exponentially with both states and actions and therefore in large problem domains learning a state-action value function can be infeasible [Kaelbling *et al.*, 1996]. Function approximation techniques have been proposed as a method for generalizing state-action value functions in large state spaces, however, this is still very much an open research problem. Conversely, PGRL methods work by starting with an initial parameterized policy which defines the probability of executing a given action in a given state. Typically, far fewer parameters are used to define the policy than there are states, thus directly addressing the need for generalization in large state spaces. PGRL algorithms work by calculating a gradient along a direction of increased reward in parameter space, and then incrementally modifying the policy parameters. PGRL algorithms are theoretically guaranteed to converge to only locally optimal policies, whereas VFRL algorithms can find globally optimal solutions. However, in practice it is usually not feasible to converge to globally optimal solutions in large problem domains in any case. Furthermore, the computational cost of PGRL algorithms grows linearly in the number of parameters used to define the policy, which is in sharp contrast to the exponential growth associated with VFRL algorithms. 2) Because PGRL algorithms start with a parameterized policy, it is relatively simple to choose a policy which incorporates prior knowledge via an appropriate choice of the parametric form of the policy. For example, if we know that in certain states only specific actions are possible, we can choose a policy parameterization which reflects this. The use of prior knowl-

edge in VFRL algorithms is not as easily realized. Finally, 3) many real problem domains are only partially observable (i.e. the agent's current state cannot be directly observed, but rather must be inferred through observations taken over many time steps), and VFRL algorithms are known to be difficult to implement in such domains. Conversely, PGRL algorithms have been shown to work effectively in partially observable problem domains [Peshkin *et al.*, 2000].

2 Theoretical Framework

2.1 MDP Assumptions

We make the usual Markov Decision Process (MDP) assumptions and for simplicity (and without loss of generality), we assume that the agent interacts with the environment in a series of episodes of duration T. The agent's state at time $t \in \{1, 2, ..., T\}$ is given by $s_t \in S$, $S \subseteq \Re^d$. Note that the state space can either be discrete or continuous. At each time step the agent chooses from one of $M > 1$ discrete actions $a_t \in A$. The dynamics of the environment are characterized by transition probabilities $P^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$.

Because we are using a policy gradient algorithm, we further assume that the policy followed by the agent is characterized by a parameter vector $\Theta = (\theta_1, ..., \theta_K) \in \Re^K$, and the probability that the agent executes action a in state s is given by $\pi(s, a; \Theta) = Pr\{a_t = a | s_t = s; \Theta\}$, $\forall s \in S, a \in A$ [Sutton *et al.*, 2000]. In addition we assume that $\pi(s, a; \Theta)$ is differentiable with respect to Θ.

2.2 Primary Reinforcer

We assume the primary reward the agent receives at time t is given by $r_t \in \Re$. The agent's goal is to optimize either the average reward function given by:

$$\rho(\Theta) = E\left\{ \frac{1}{T} \sum_{t=1}^{T} r_t \middle| \Theta \right\} \tag{1}$$

or the discounted reward function given by:

$$\rho(\Theta) = E\left\{ \sum_{t=1}^{T} \gamma^{t-1} r_t \middle| s_0, \Theta \right\} \tag{2}$$

where $0 < \gamma < 1$ is a discount factor and s_0 is some initial starting state. Note that both reward functions are assumed to be a function of the agent's policy which is parameterized by Θ.

2.3 Secondary Reinforcers

At each time t the agent senses one of N stimuli denoted by $((sr)^1_t, ..., (sr)^N_t)$, where $(sr)^n_t \in \Re$, $\forall n = \{1, 2, ..., N\}$. These N stimuli constitute the agent's secondary reinforcers, and the average expected stimuli over an episode for each reinforcer, here termed *secondary reinforcer frequencies*, is given by:

$$F_n(\Theta) = E\left\{ \frac{1}{T} \sum_{t=1}^{T} (sr)^n_t \middle| \Theta \right\} \tag{3}$$

$\forall n = \{1, 2, ..., N\}$. As with the primary reward functions (1) and (2), these secondary reward stimulus functions are a function of the agent's policy which is defined by the parameters Θ.

2.4 SRPG Algorithm

Let i indicate the number of times each step in the algorithm has been executed, and $\Phi^i = \left(\phi^i_1, ..., \phi^i_N\right)$ be the current desired secondary reinforcer frequencies corresponding to $\mathbf{F}(\Theta) = \left(F^i_1(\Theta), ..., F^i_N(\Theta)\right)$. The Secondary Reinforcers Policy Gradient (SRPG) algorithm consists of the following two steps:

STEP 1: *Learn a policy that achieves the currently desired frequency of secondary reinforcers* : In this step the agent establishes these secondary reinforcer frequencies by learning a set of policy parameters Θ_i that solve the system of N nonlinear equations defined by:

$$\mathbf{F}(\Theta_i) = \Phi^i \tag{4}$$

STEP 2: *Calculate the gradient of increased primary reward in secondary reinforcers space and update desired secondary reinforcer frequencies*: Estimate the gradient of the primary reward function $\rho(\Theta_i)$ with respect to the desired secondary reinforcer frequencies Φ^i:

$$\frac{\partial \rho(\Theta_i)}{\partial \Phi^i} \tag{5}$$

Given this estimate, the next estimate for the secondary reinforcer frequencies, Φ^{i+1}, is given by:

$$\Phi^{i+1} = \Phi^i + \alpha \frac{\partial \rho(\Theta_i)}{\partial \Phi^i} \tag{6}$$

where $\alpha \in \Re$ is a small positive step size. Increment i and goto *STEP 1*.

Note that in *STEP 1* of the algorithm the agent does not need to observe the primary reward r_t. Instead the agent must solve the nonlinear system of equations (4). This can be done using a number of different algorithms [Press *et al.*, 1988]; here we briefly describe one such procedure. The algorithm used in this paper is to first estimate the gradient:

$$\frac{\partial \left(\mathbf{F}\left(\Theta^j\right) - \Phi^i\right)}{\partial \Theta^j} = -\frac{1}{2} \sum_{n=1}^{N} \left(F_n\left(\Theta^j\right) - \varphi^i_n\right) \frac{\partial F_n\left(\Theta^j\right)}{\partial \Theta^j}$$

and then use the update equation:

$$\Theta^{j+1} = \Theta^j + \beta \frac{\partial \left(\mathbf{F}\left(\Theta^j\right) - \Phi^i\right)}{\partial \Theta^j} \tag{7}$$

where $\beta \in \Re$ is a small positive step size. Assuming that the algorithm doesn't run into a local minimum, Θ^j will converge to Θ^i as j becomes large. See Section 4 for a discussion of what happens when the algorithm encounters a local minimum. Note that the algorithm requires estimates of:

$$\frac{\partial F_n\left(\Theta^j\right)}{\partial \Theta^j}$$

for $n = (1, ..., N)$. These estimates can be made using standard policy gradient algorithms [Williams, 1987; 1992;

Sutton *et al.*, 2000; Konda and Tsitsiklis, 2000; Baxter and Bartlett, 2000]. For example, if we use the formulation defined in [Sutton *et al.*, 2000], then

$$\frac{\partial F_n\left(\Theta^j\right)}{\partial \Theta^j} = \sum_s d^\pi\left(s\right) \sum_a \frac{\partial \pi(s,a;\theta)}{\partial \theta} Q^\pi_{F_n}(s,a) \qquad (8)$$

where $d^\pi\left(s\right)$ is the stationary distribution of states under π and $Q^\pi_{F_n}(s,a)$ is given by:

$$Q^\pi_{F_n}(s,a) = E\left\{ \frac{1}{T} \sum_{k=1}^{T} (sr)^n_{t+k} \,\middle|\, s_t = s, a_t = a, \pi \right\}$$

In *STEP 2* of the algorithm the agent must numerically calculate the gradient (5). This is numerically estimated $\forall n = 1, ..., N$ using the following:

$$\frac{\partial \rho\left(\Theta_i\right)}{\partial \phi^i_n} = \lim_{\delta_n \to 0} \frac{\rho\left(\Theta|\left(\mathbf{F}\left(\Theta^j\right) \equiv \Phi^i + \Delta_n\right)\right) - \rho\left(\Theta_i\right)}{\delta_n} \qquad (9)$$

where $\Delta_n \in \Re^N$, such that every element of the vector is zero except for element n. Therefore (5) is estimated using small perturbations (i.e. Δ_n) in each of the elements of Φ^i, solving the system $\mathbf{F}(\Theta^*_i) = \Phi^i + \Delta_n$ for Θ^*_i as outlined above, and then observing the associated $\rho\left(\Theta|\left(\mathbf{F}\left(\Theta^j\right) \equiv \Phi^i + \Delta_n\right)\right)$. Thus in *STEP 2* the agent must observe the primary reward function ρ at least $N+1$ times (once for $\rho\left(\Theta_i\right)$ and once for each of the N secondary stimuli for $\rho\left(\Theta|\left(\mathbf{F}\left(\Theta^j\right) \equiv \Phi^i + \Delta_n\right)\right)$.

2.5 Convergence Results

We present two theorems. The first establishes the conditions under which the SRPG algorithm converges.

Theorem 1: *Let* $\alpha_i \in \Re$, $\forall i = 0, 1, ...,$ *such that* $\lim_{i\to\infty} \alpha_i = 0$ *and* $\sum_i \alpha_i = \infty$. *Assume that* $\max_{s,a,k,j,\theta} \left|\frac{\partial^2 \pi(s,a;\Theta)}{\partial \theta_k \partial \theta_j}\right| < K < \infty$, $\max_{s,a,k,j,\phi} \left|\frac{\partial^2 \rho(\Theta_i)}{\partial \phi^i_k \partial \phi^i_j}\right| < K < \infty$ *and that* Θ_i *exists such that (4) is satisfied. Then, the sequence of* Θ_i *defined by the SRPG algorithm converges such that for any MDP with bounded primary reward* $\lim_{i\to\infty} \frac{\partial \rho(\Theta_i)}{\partial \Phi^i} = 0$.

Proof: The necessary condition on the bound $\frac{\partial^2 F(\Theta_i)}{\partial \theta_k \partial \theta_j}$ is established by the bound on $\frac{\partial^2 \pi(s,a;\Theta)}{\partial \theta_k \partial \theta_j}$ due to equation (8). The remainder of the proof then follows directly from Proposition 3.5 on page 96 of [Bertsekas and Tsitsiklis, 1996]. □

The second theorem proves that the rate of convergence of the gradient estimate (9) (with respect to the number of primary reinforcers observed) is independent of the size of the agent's state space.

Theorem 2: $\forall n = 1, ..., N$ *and* $\Delta_n \in \Re^N$ *assume that there exists a finite* $k \in \Re$ $\left|\rho\left(\Theta|\left(\mathbf{F}\left(\Theta^j\right) \equiv \Phi^i + \Delta_n\right)\right) - \rho\left(\Theta_i\right)\right| \leq k\left\|\Delta_n\right\|$ *(i.e. Lipschitz smoothness condition). Assume also that the variance in the primary reinforcer estimate* ρ *is* σ^2_ρ. *Then, for any arbitrary small positive* $\epsilon \in \Re$, *there exists a* Δ_n *such that*

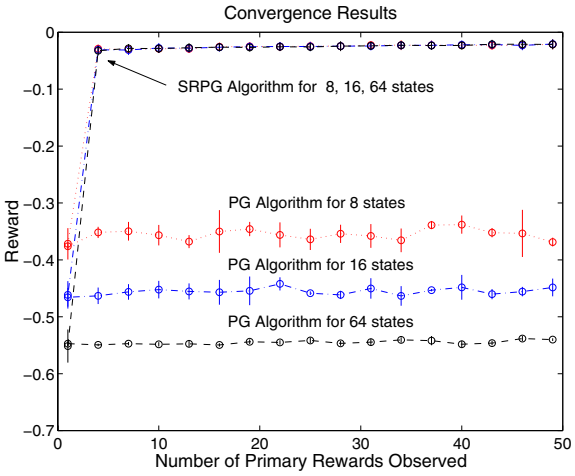


Figure 1: Convergence results, as a function the number of observations of the primary reinforcer, for MDP problems with 8, 16, and 64 states are given.

$$\left|\frac{\rho\left(\Theta|\left(\mathbf{F}\left(\Theta^j\right) \equiv \Phi^i + \Delta_n\right)\right) - \rho\left(\Theta_i\right)}{\delta_n} - \frac{\partial \rho\left(\Theta_i\right)}{\partial \phi^i_n}\right| \leq \varepsilon \qquad (10)$$

Further, assume that each observation of ρ *is independent, then the variance in the gradient is given by:*

$$V\left[\frac{\rho\left(\Theta|\left(\mathbf{F}\left(\Theta^j\right) \equiv \Phi^i + \Delta_n\right)\right) - \rho\left(\Theta_i\right)}{\delta_n}\right] = \frac{2}{M\delta^2_n}\sigma^2_\rho \qquad (11)$$

where M *is the number of times* $\rho\left(\Theta|\left(\mathbf{F}\left(\Theta^j\right) \equiv \Phi^i + \Delta_n\right)\right)$ *and* $\rho\left(\Theta_i\right)$ *are observed.*

Proof: Equation (10) is satisfied by choosing δ_n such that $k\left\|\Delta_n\right\| \leq \epsilon$. Equation (11) is obtained by noting that if $U = a_1 Y_1 + a_2 Y_2$ and if Y_1 and Y_2 are independent, then $V(U) = a^2_1 V(Y_1) + a^2_2 V(Y_2)$. Further, if $V(Y) = \sigma^2_Y$ and $\hat{Y}$ is observations of Y, then if Y is observed M times, then $\hat{Y} = \frac{\sigma^2_Y}{M}$. □

3 Experimental Results

We implemented MDP examples with 8, 16 and 64 discrete states respectively. The MDPs were fully connected and therefore in the 8 state example the agent had 8 actions to choose from in each state, in the 16 state example the agent had 16 actions in each state, and so on. The actions are chosen using a Boltzmann distribution as follows:

$$\pi\left(s = i, a = j; \Theta\right) = \frac{e^{\theta_{ij}}}{\sum_{k=1}^{d} e^{\theta_{ik}}}$$

where d is the total number of states. The transition probabilities of the MDP are as follows: if action $a_t = j$ is chosen in state s_t at time t, then $p(s_{t+1} = j | s_t, a_t = j) = 0.95$. The agent interacts with the MDP in a series of episodes lasting 10,000 time steps each, always starting in state $s = 1$.

The following reward is given to the agent at the end of each episode:

$$\rho(\Theta) = r = -\left(\left(\mathrm{N}_{s=1} - 0.7\right)^2 + \left(\mathrm{N}_{s=2} - 0.3\right)^2\right)$$

where $\mathrm{N}_{s=j} = \frac{\text{no of visits to state } j}{10{,}000}$. Thus, in all three examples, the agent achieves maximum reward if it spends 70% of the time in state $s = 1$ and 30% of the time in state $s = 2$. Initially, the values for all the policy parameters θ_{sa} are set to 1.0.

Two secondary reinforcers are used by the SRPG algorithm: $\phi_1 = \mathrm{N}_{s=1}$ and $\phi_2 = \mathrm{N}_{s=2}$. The agent starts with the desired secondary reinforcer frequencies $\phi_1^0 = 0.5$ and $\phi_2^0 = 0.5$ in STEP 1 of the SRPG algorithm. The SRPG algorithm learning rates are set to $\alpha = \beta = 0.01$. Figure 1 shows learning curves averaged over 10 runs with error bars indicating standard deviation. Note that, as predicted by **Theorem 1**, the convergence of the algorithm as a function of the number of observed primary rewards is not affected by the number of states in the MDP.

The rate of convergence of the SRPG algorithm was compared to the the PG algorithm proposed in [Sutton *et al.*, 2000]:

$$\frac{\partial \rho\left(\Theta^j\right)}{\partial \Theta^j} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a;\theta)}{\partial \theta} Q^\pi(s,a)$$

where $d^\pi(s)$ is the stationary distribution of states under π (estimated after each episode by counting the number of times each state is visited) and $Q^\pi(s,a)$ is given by:

$$Q^\pi(s,a) = E\left\{ \frac{1}{T} \sum_{k=1}^{T} r_{t+k} \,\middle|\, s_t = s, a_t = a, \pi \right\}$$

A typical learning curve, as a function of the number of observations of the primary reinforcer, for this PG algorithm for the 8 state MPD is given in Figure 2 (as with the SRPG algorithm the learning step size was set to $\alpha = 0.01$). Note that convergence took approximately 5000 observations of the primary reward function ρ. Convergence of the PG algorithm for the 16 state example took about 28,000 observations of ρ. For the 64 state example, convergence was not observed after 1,000,000 observations of ρ, at which time the algorithm was stopped.

Figure 1 also shows the learning curves (averaged over 10 runs with error bars showing standard deviation) for the PG algorithm after the first 50 observations of the primary reward ρ. Note that although the SRPG algorithm converges after about 50 runs for all three examples, the PG algorithm does not appreciably improve the reward received by agent over this number of observations of ρ.

4 Conclusion and Discussion

In large problem domains, prior knowledge is essential for an agent to effectively learn via reinforcement reward. However, how domain specific knowledge should be added to RL algorithms to the agent's advantage is still an open research problem. Secondary reinforcers are stimuli that are directly related to a primary reinforcement, and thus their relative reward properties can be measured with respect to agent's global reward function. In this paper we show that secondary reinforcers are a fruitful source of domain specific knowledge which can be effectively used to guide search. In particular, if the primary reinforcement is observed only rarely, while secondary reinforcers are more readily observable, the rate of convergence of RL algorithms, with respect to the number of observations of a primary reward, can be significantly improved.

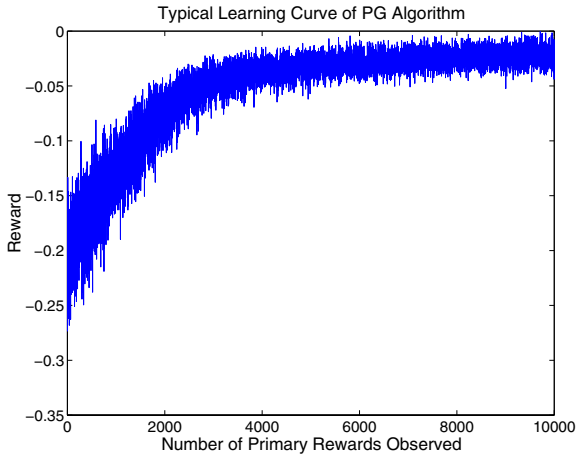


Figure 2: A typical learning curve for the PG algorithm (as a function the number of observations of the primary reinforcer) on an 8 state MDP problem.

However, not all secondary reinforcers should be blindly maximized. Some are more important than others, and in order to make full use of secondary reinforcers, an agent should control the relative amounts of this environmental feedback. This paper presents a new policy gradient algorithm, called Secondary Reinforcer Policy Gradient (SRPG), that iterates in the space of secondary reinforcer stimuli to a locally optimal policy. The algorithm consists of two steps. The first step starts with an assumption of the relative amounts of secondary reinforcer stimuli that the agent will need to establish to achieve good primary reward. In this first step the agent learns a policy which achieves these relative amounts of stimuli using a standard policy gradient RL algorithm. A key property of SRPG is that a primary reward need not be observed during this step of the algorithm, and therefore the convergence of algorithm is not slowed because the primary reward is rarely observed. The second step of the algorithm involves the calculation of a gradient in the direction increased primary reward, in the space secondary reinforcer stimuli. If there are N secondary reinforcers, then step two of the algorithm requires that the primary reinforcer be sampled $N + 1$ times. However, we present theory showing that the rate of convergence of the gradient estimate is independent of the size of the state space, making the algorithm viable on high dimensional problems. This estimated gradient is used to update the relative desired amounts of secondary reinforcers, and step one of the algorithm is then executed to establish a policy to achieve these relative amounts of stim-

uli, thus improving the primary reward. Step one and two are repeated until the policy converges to a local optimum.

Experimental results are presented which show that the SRPG algorithm can improve the convergence of Policy Gradient (PG) algorithms, requiring many orders of magnitude fewer observations of the primary reward to converge. In addition, we present theory showing that the SRPG algorithm converges to a locally optimal policy (with respect to the primary reward) in the space of secondary reinforcers. However, in order for this theorem to hold, step one of the algorithm must be able to establish a policy that achieves the specified relative amounts of secondary reinforcer stimuli. The algorithm used in this paper to solve this nonlinear system of equations is based on gradient descent, and is therefore prone to convergence to local minimum. However, in the experimental results presented here, the SRPG algorithm converged to close to optimal primary reward, even when local minima were encountered in step one of the algorithm. The reason for this is that these local minima sufficiently improved the primary reward, thus allowing the overall algorithm to continue to converge. Therefore, establishing a more relaxed set of conditions under which the SRPG algorithm will converge is an open research topic.

Further open research questions include the use of secondary reinforcers within the actor-critic reinforcement learning framework [Sutton and Barto, 1998]. Actor critic algorithms combine policy gradient and value function methods and thus can often achieve successful policies when either method alone would fail. Such algorithms may further benefit from domain specific knowledge derived from secondary reinforcers. Finally, an important open research topic is, given a large list of potential secondary reinforcers, formulate an algorithm that finds a small subset which can be used to improve RL algorithms.

Acknowledgements

Thanks to Jane Mulligan for useful discussions. This work was funded by the GRASP Lab, the IRCS at the University of Pennsylvania, and by the DARPA ITO MARS grant no. DABT63-99-1-0017.

References

[Baird and Moore, 1999] L. Baird and A. W. Moore. Gradient descent for general reinforcement learning. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 11, Cambridge, MA, 1999. MIT Press.

[Baxter and Bartlett, 2000] Jonathan Baxter and Peter L. Bartlett. Reinforcement learning in POMDP's via direct gradient ascent. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML'2000)*, pages 41–48, Stanford University, CA, June 2000.

[Bertsekas and Tsitsiklis, 1996] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[Grudic and Ungar, 2000a] G. Z. Grudic and L. H. Ungar. Localizing policy gradient estimates to action transitions. In *Proceedings of the Seventeenth International Conference on Machine Learning*, volume 17, pages 343–350. Morgan Kaufmann, June 29 - July 2 2000.

[Grudic and Ungar, 2000b] G. Z. Grudic and L. H. Ungar. Localizing search in reinforcement learning. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, volume 17, pages 590–595. Menlo Park, CA: AAAI Press / Cambridge, MA: MIT Press, July 30 - August 3 2000.

[Kaelbling *et al.*, 1996] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Konda and Tsitsiklis, 2000] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In S. A. Solla, T. K. Leen, and K.-R. Mller, editors, *Advances in Neural Information Processing Systems*, volume 12, Cambridge, MA, 2000. MIT Press.

[Minsky, 1963] Marvin Minsky. Steps toward artificial intelligence. In E. A. Feigenbaum and J. Feldman, editors, *Computers and Thought*, pages 406–450. McGraw-Hill Book Company, New York, N.Y., 1963.

[Peshkin *et al.*, 2000] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Pack Kaelbling. Learning to cooperate via policy search. In *Proceedings of the Sixteenth International Conference on Uncertainty in Artificial Intelligence (UAI 2000)*, Stanford University, CA, June 2000.

[Press *et al.*, 1988] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, New York NY, 1988.

[Sutton and Barto, 1990] R. S. Sutton and A. G. Barto. Time-derivative models of Pavlovian reinforcement. In M. Gabriel and J. Moore, editors, *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, pages 497–537. MIT Press, 1990.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[Sutton *et al.*, 2000] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. A. Solla, T. K. Leen, and K.-R. Mller, editors, *Advances in Neural Information Processing Systems*, volume 12, Cambridge, MA, 2000. MIT Press.

[Williams, 1987] R. J. Williams. A class of gradient-estimating algorithms for reinforcement learning in neural networks. In *Proceedings of the IEEE First International Conference on Neural Networks*, San Diego, CA, 1987.

[Williams, 1992] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

MACHINE LEARNING AND DATA MINING

MACHINE LEARNING AND DATA MINING

The Foundations of Cost-Sensitive Learning

Charles Elkan
Department of Computer Science and Engineering 0114
University of California, San Diego
La Jolla, California 92093-0114
elkan@cs.ucsd.edu

Abstract

This paper revisits the problem of optimal learning and decision-making when different misclassification errors incur different penalties. We characterize precisely but intuitively when a cost matrix is reasonable, and we show how to avoid the mistake of defining a cost matrix that is economically incoherent. For the two-class case, we prove a theorem that shows how to change the proportion of negative examples in a training set in order to make optimal cost-sensitive classification decisions using a classifier learned by a standard non-cost-sensitive learning method. However, we then argue that changing the balance of negative and positive training examples has little effect on the classifiers produced by standard Bayesian and decision tree learning methods. Accordingly, the recommended way of applying one of these methods in a domain with differing misclassification costs is to learn a classifier from the training set as given, and then to compute optimal decisions explicitly using the probability estimates given by the classifier.

1 Making decisions based on a cost matrix

Given a specification of costs for correct and incorrect predictions, an example should be predicted to have the class that leads to the lowest expected cost, where the expectation is computed using the conditional probability of each class given the example. Mathematically, let the (i, j) entry in a cost matrix C be the cost of predicting class i when the true class is j. If $i = j$ then the prediction is correct, while if $i \neq j$ the prediction is incorrect. The optimal prediction for an example x is the class i that minimizes

$$L(x, i) = \sum_j P(j|x)C(i, j). \quad (1)$$

Costs are not necessarily monetary. A cost can also be a waste of time, or the severity of an illness, for example.

For each i, $L(x, i)$ is a sum over the alternative possibilities for the true class of x. In this framework, the role of a learning algorithm is to produce a classifier that for any example x can estimate the probability $P(j|x)$ of each class j being the true class of x. For an example x, making the prediction i means acting as if i is the true class of x. The essence of cost-sensitive decision-making is that it can be optimal to act as if one class is true even when some other class is more probable. For example, it can be rational not to approve a large credit card transaction even if the transaction is most likely legitimate.

1.1 Cost matrix properties

A cost matrix C always has the following structure when there are only two classes:

	actual negative	actual positive
predict negative	$C(0,0) = c_{00}$	$C(0,1) = c_{01}$
predict positive	$C(1,0) = c_{10}$	$C(1,1) = c_{11}$

Recent papers have followed the convention that cost matrix rows correspond to alternative predicted classes, while columns correspond to actual classes, i.e. row/column = i/j = predicted/actual.

In our notation, the cost of a false positive is c_{10} while the cost of a false negative is c_{01}. Conceptually, the cost of labeling an example incorrectly should always be greater than the cost of labeling it correctly. Mathematically, it should always be the case that $c_{10} > c_{00}$ and $c_{01} > c_{11}$. We call these conditions the "reasonableness" conditions.

Suppose that the first reasonableness condition is violated, so $c_{00} \geq c_{10}$ but still $c_{01} > c_{11}$. In this case the optimal policy is to label all examples positive. Similarly, if $c_{10} > c_{00}$ but $c_{11} \geq c_{01}$ then it is optimal to label all examples negative. We leave the case where both reasonableness conditions are violated for the reader to analyze.

Margineantu [2000] has pointed out that for some cost matrices, some class labels are never predicted by the optimal policy as given by Equation (1). We can state a simple, intuitive criterion for when this happens. Say that row m dominates row n in a cost matrix C if for all j, $C(m, j) \geq C(n, j)$. In this case the cost of predicting n is no greater than the cost of predicting m, regardless of what the true class j is. So it is optimal never to predict m. As a special case, the optimal prediction is always n if row n is dominated by all other rows in a cost matrix. The two reasonableness conditions for a two-class cost matrix imply that neither row in the matrix dominates the other.

Given a cost matrix, the decisions that are optimal are unchanged if each entry in the matrix is multiplied by a positive constant. This scaling corresponds to changing the unit of

account for costs. Similarly, the decisions that are optimal are unchanged if a constant is added to each entry in the matrix. This shifting corresponds to changing the baseline away from which costs are measured. By scaling and shifting entries, any two-class cost matrix that satisfies the reasonableness conditions can be transformed into a simpler matrix that always leads to the same decisions:

0	c'_{01}
1	c'_{11}

where $c'_{01} = (c_{01} - c_{00})/(c_{10} - c_{00})$ and $c'_{11} = (c_{11} - c_{00})/(c_{10} - c_{00})$. From a matrix perspective, a 2x2 cost matrix effectively has two degrees of freedom.

1.2 Costs versus benefits

Although most recent research in machine learning has used the terminology of costs, doing accounting in terms of benefits is generally preferable, because avoiding mistakes is easier, since there is a natural baseline from which to measure all benefits, whether positive or negative. This baseline is the state of the agent before it takes a decision regarding an example. After the agent has made the decision, if it is better off, its benefit is positive. Otherwise, its benefit is negative.

When thinking in terms of costs, it is easy to posit a cost matrix that is logically contradictory because not all entries in the matrix are measured from the same baseline. For example, consider the so-called German credit dataset that was published as part of the Statlog project [Michie *et al.*, 1994]. The cost matrix given with this dataset is as follows:

	actual bad	actual good
predict bad	0	1
predict good	5	0

Here examples are people who apply for a loan from a bank. "Actual good" means that a customer would repay a loan while "actual bad" means that the customer would default. The action associated with "predict bad" is to deny the loan. Hence, the cashflow relative to any baseline associated with this prediction is the same regardless of whether "actual good" or "actual bad" is true. In every economically reasonable cost matrix for this domain, both entries in the "predict bad" row must be the same.

Costs or benefits can be measured against any baseline, but the baseline must be fixed. An opportunity cost is a foregone benefit, i.e. a missed opportunity rather than an actual penalty. It is easy to make the mistake of measuring different opportunity costs against different baselines. For example, the erroneous cost matrix above can be justified informally as follows: "The cost of approving a good customer is zero, and the cost of rejecting a bad customer is zero, because in both cases the correct decision has been made. If a good customer is rejected, the cost is an opportunity cost, the foregone profit of 1. If a bad customer is approved for a loan, the cost is the lost loan principal of 5."

To see concretely that the reasoning in quotes above is incorrect, suppose that the bank has one customer of each of the four types. Clearly the cost matrix above is intended to imply that the net change in the assets of the bank is then -4. Alternatively, suppose that we have four customers who receive loans and repay them. The net change in assets is then +4. Regardless of the baseline, any method of accounting should give a difference of 8 between these scenarios. But with the erroneous cost matrix above, the first scenario gives a total cost of 6, while the second scenario gives a total cost of 0.

In general the amount in some cells of a cost or benefit matrix may not be constant, and may be different for different examples. For example, consider the credit card transactions domain. Here the benefit matrix might be

	fraudulent	legitimate
refuse	\$20	$-$\$20
approve	$-x$	$0.02x$

where x is the size of the transaction in dollars. Approving a fraudulent transaction costs the amount of the transaction because the bank is liable for the expenses of fraud. Refusing a legitimate transaction has a non-trivial cost because it annoys a customer. Refusing a fraudulent transaction has a non-trivial benefit because it may prevent further fraud and lead to the arrest of a criminal. Research on cost-sensitive learning and decision-making when costs may be example-dependent is only just beginning [Zadrozny and Elkan, 2001a].

1.3 Making optimal decisions

In the two-class case, the optimal prediction is class 1 if and only if the expected cost of this prediction is less than or equal to the expected cost of predicting class 0, i.e. if and only if

$$P(j = 0|x)c_{10} + P(j = 1|x)c_{11} \leq P(j = 0|x)c_{00} + P(j = 1|x)c_{01}$$

which is equivalent to

$$(1 - p)c_{10} + pc_{11} \leq (1 - p)c_{00} + pc_{01}$$

given $p = P(j = 1|x)$. If this inequality is in fact an equality, then predicting either class is optimal.

The threshold for making optimal decisions is p^* such that

$$(1 - p^*)c_{10} + p^*c_{11} = (1 - p^*)c_{00} + p^*c_{01}.$$

Assuming the reasonableness conditions the optimal prediction is class 1 if and only if $p \geq p^*$. Rearranging the equation for p^* leads to the solution

$$p^* = \frac{c_{10} - c_{00}}{c_{10} - c_{00} + c_{01} - c_{11}} \quad (2)$$

assuming the denominator is nonzero, which is implied by the reasonableness conditions. This formula for p^* shows that any 2x2 cost matrix has essentially only one degree of freedom from a decision-making perspective, although it has two degrees of freedom from a matrix perspective. The cause of the apparent contradiction is that the optimal decision-making policy is a nonlinear function of the cost matrix.

2 Achieving cost-sensitivity by rebalancing

In this section we turn to the question of how to obtain a classifier that is useful for cost-sensitive decision-making.

Standard learning algorithms are designed to yield classifiers that maximize accuracy. In the two-class case, these classifiers implicitly make decisions based on the probability

threshold 0.5. The conclusion of the previous section was that we need a classifier that given an example x, says whether or not $P(j = 1|x) \geq p^*$ for some target threshold p^* that in general is different from 0.5. How can a standard learning algorithm be made to produce a classifier that makes decisions based on a general p^*?

The most common method of achieving this objective is to rebalance the training set given to the learning algorithm, i.e. to change the the proportion of positive and negative training examples in the training set. Although rebalancing is a common idea, the general formula for how to do it correctly has not been published. The following theorem provides this formula.

Theorem 1: To make a target probability threshold p^* correspond to a given probability threshold p_0, the number of negative examples in the training set should be multiplied by

$$\frac{p^*}{1-p^*}\frac{1-p_0}{p_0}. \qquad \blacksquare$$

While the formula in Theorem 1 is simple, the proof of its correctness is not. We defer the proof until the end of the next section.

In the special case where the threshold used by the learning method is $p_o = 0.5$ and $c_{00} = c_{11} = 0$, the theorem says that the number of negative training examples should be multiplied by $p^*/(1-p^*) = c_{10}/c_{01}$. This special case is used by Breiman *et al.* [1984].

The directionality of Theorem 1 is important to understand. Suppose we have a learning algorithm L that yields classifiers that make predictions based on a probability threshold p_0. Given a training set S and a desired probability threshold p^*, the theorem says how to create a training set S' by changing the number of negative training examples such that L applied to S' gives the desired classifier.

Theorem 1 does not say in what way the number of negative examples should be changed. If a learning algorithm can use weights on training examples, then the weight of each negative example can be set to the factor given by the theorem. Otherwise, we must do oversampling or undersampling. Oversampling means duplicating examples, and undersampling means deleting examples.

Sampling can be done either randomly or deterministically. While deterministic sampling can reduce variance, it risks introducing bias, if the non-random choice of examples to duplicate or eliminate is correlated with some property of the examples. Undersampling that is deterministic in the sense that the fraction of examples with each value of a certain feature is held constant is often called stratified sampling.

It is possible to change the number of positive examples instead of or as well as changing the number of negative examples. However in many domains one class is rare compared to the other, and it is important to keep all available examples of the rare class. In these cases, if we call the rare class the positive class, Theorem 1 says directly how to change the number of common examples without discarding or duplicating any of the rare examples.

3 New probabilities given a new base rate

In this section we state and prove a theorem of independent interest that happens also to be the tool needed to prove Theorem 1. The new theorem answers the question of how the predicted class membership probability of an example should change in response to a change in base rates. Suppose that $p = P(j = 1|x)$ is correct for an example x, if x is drawn from a population with base rate $b = P(j = 1)$ positive examples. But suppose that in fact x is drawn from a population with base rate b'. What is $p' = P'(j = 1|x)$?

We make the assumption that the shift in base rate is the only change in the population to which x belongs. Formally, we assume that within the positive and negative subpopulations, example probabilities are unchanged: $P'(x|j = 1) = P(x|j = 1)$ and $P'(x|j = 0) = P(x|j = 0)$. Given these assumptions, the following theorem shows how to compute p' as a function of p, b, and b'.

Theorem 2: In the context just described,

$$p' = b'\frac{p - pb}{b - pb + b'p - bb'}.$$

Proof: Using Bayes' rule, $p = P(j = 1|x)$ is

$$\frac{P(x|j=1)P(j=1)}{P(x)} = \frac{P(x|j=1)b}{P(x)}.$$

Because $j = 1$ and $j = 0$ are mutually exclusive, $P(x)$ is

$$P(x|j=1)P(j=1) + P(x|j=0)P(j=0).$$

Let $c = P(x|j = 1)$, let $d = P(x|j = 0)$, and let $e = d/c$. Then

$$p = \frac{cb}{cb + d(1-b)} = \frac{b}{b + e(1-b)}.$$

Similarly,

$$p' = \frac{b'}{b' + e(1-b')}.$$

Now we can solve for e as a function of p and b. We have $pb + pe(1-b) = b$ so $e = (b - pb)/(p - pb)$. Then the denominator for p' is

$$b' + e(1-b') = b' + \frac{b-pb}{p-pb} - b'\frac{b-pb}{p-pb}$$
$$= \frac{b-pb}{p-pb} + b'\left(1 - \frac{b-pb}{p-pb}\right) = \frac{b - pb + b'p - bb'}{p-pb}.$$

Finally we have

$$p' = b'\frac{p-pb}{b-pb+b'p-bb'}. \qquad \blacksquare$$

It is important to note that Theorem 2 is a statement about true probabilities given different base rates. The proof does not rely on how probabilities may be estimated based on some learning process. In particular, the proof does not use any assumptions of independence or conditional independence, as made for example by a naive Bayesian classifier.

If a classifier yields estimated probabilities $\hat{p}$ that we assume are correct given a base rate b, then Theorem 2 lets us

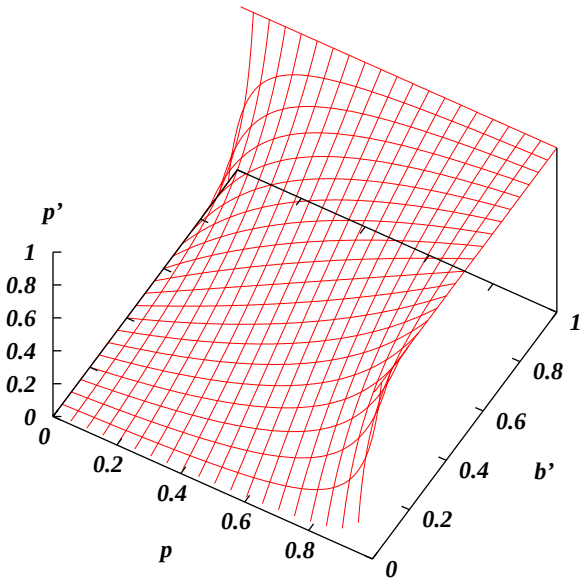


Figure 1: p' as a function of p and b, when $b' = 0.5$.

compute estimated probabilities $\hat{p}'$ that are correct given a different base rate b'. From this point of view, the theorem has a remarkable aspect. It lets us use a classifier learned from a training set drawn from one probability distribution on a test set drawn from a different probability distribution. The theorem thus relaxes one of the most fundamental assumptions of almost all research on machine learning, that training and test sets are drawn from the same population.

The insight in the proof is the introduction of the variable e that is the ratio of $P(x|j = 0)$ and $P(x|j = 1)$. If we try to compute the actual values of these probabilities, we find that we have more variables to solve for than we have simultaneous equations. Fortunately, all we need to know for any particular example x is the ratio e.

The special case of Theorem 2 where $p' = 0.5$ was recently worked out independently by Weiss and Provost [2001]. The case where $b = 0.5$ is also interesting. Suppose that we do not know the base rate of positive examples at the time we learn a classifier. Then it is reasonable to use a training set with $b = 0.5$. Theorem 2 says how to compute probabilities p' later that are correct given that the population of test examples has base rate b'. Specifically,

$$p' = b' \frac{p - p/2}{1/2 - p/2 + b'p - b'/2} = \frac{p}{p + (1-p)(1-b')/b'}.$$

This function of p and b' is plotted in Figure 1.

Using Theorem 2 as a lemma, we can now prove Theorem 1 with a slight change of notation.

Theorem 1: To make a target probability threshold p correspond to a given probability threshold p', the number of negative training examples should be multiplied by

$$\frac{p}{1-p}\frac{1-p'}{p'}.$$

Proof: We want to compute an adjusted base rate b' such that for a classifier trained using this base rate, an estimated probability p' corresponds to a probability p for a classifier trained using the base rate b.

We need to compute the adjusted b' as a function of b, p, and p'. From the proof of Theorem 2, $p'b - p'pb + p'b'p - p'bb' = b'p - b'pb$. Collecting all the b' terms on the left, we have $b'p - b'pb - b'p'p + b'p'b = p'b - p'pb$, which gives that the adjusted base rate should be

$$b' = \frac{p'b - p'pb}{p - pb - p'p + p'b} = \frac{p'b(1-p)}{p - pb - p'p + p'b}.$$

Suppose that $b = 1/(1+n)$ and $b' = 1/(1+n')$ so the number of negative training examples should be multiplied by n'/n to get the adjusted base rate b'. We have that $n' = (1-b')/b'$ is

$$\frac{p - pb - p'p + p'b - p'b + p'bp}{p - pb - p'p + p'b}\frac{p - pb - p'p + p'b}{p'b(1-p)}$$
$$= \frac{p(1-b) - p'p(1-b)}{p'b(1-p)} = \frac{p(1-b)(1-p')}{p'b(1-p)}.$$

Therefore

$$\frac{n'}{n} = \frac{p(1-b)(1-p')}{p'b(1-p)}\frac{b}{1-b} = \frac{p(1-p')}{p'(1-p)}.$$ ∎

Note that the effective cardinality of the subset of negative training examples must be changed in a way that does not change the distribution of examples within this subset.

4 Effects of changing base rates

Changing the training set prevalence of positive and negative examples is a common method of making a learning algorithm cost-sensitive. A natural question is what effect such a change has on the behavior of standard learning algorithms. Separately, many researchers have proposed duplicating or discarding examples when one class of examples is rare, on the assumption that standard learning methods perform better when the prevalence of different classes is approximately equal [Kubat and Matwin, 1997; Japkowicz, 2000]. The purpose of this section is to investigate this assumption.

4.1 Changing base rates and Bayesian learning

Given an example x, a Bayesian classifier applies Bayes' rule to compute the probability of each class j as $P(j|x) = P(x|j)P(j)/P(x)$. Typically $P(x|j)$ is computed by a function learned from a training set, $P(j)$ is estimated as the training set frequency of class j, and $P(x)$ is computed indirectly by solving the equation $\sum_j P(j|x) = 1$.

A Bayesian learning method essentially learns a model $P(x|j)$ of each class j separately. If the frequency of a class is changed in the training set, the only change is to the estimated base rate $P(j)$ of each class. Therefore there is little reason to expect the accuracy of decision-making with a Bayesian classifier to be higher with any particular base rates.

Naive Bayesian classifiers are the most important special case of Bayesian classification. A naive Bayesian classifier is based on the assumption that within each class, the values of the attributes of examples are independent. It is well-known that these classifiers tend to give inaccurate probability estimates [Domingos and Pazzani, 1996]. Given

an example x, suppose that a naive Bayesian classifier computes $N(x)$ as its estimate of $P(j = 1|x)$. Usually $N(x)$ is too extreme: for most x, either $N(x)$ is close to 0 and then $N(x) < P(j = 1|x)$ or $N(x)$ is close to 1 and then $N(x) > P(j = 1|x)$.

However, the ranking of examples by naive Bayesian classifiers tends to be correct: if $N(x) < N(y)$ then $P(j = 1|x) < P(j = 1|y)$. This fact suggests that given a cost-sensitive application where optimal decision-making uses the probability threshold p^*, one should empirically determine a different threshold $\bar{p}$ such that $N(x) \geq \bar{p}$ is equivalent to $P(j = 1|x) \geq p^*$. This procedure is likely to improve the accuracy of decision-making, while changing the proportion of negative examples using Theorem 1 in order to use the threshold 0.5 is not.

4.2 Decision tree growing

We turn our attention now to standard decision tree learning methods, which have two phases. In the first phase a tree is grown top-down, while in the second phase nodes are pruned from the tree. We discuss separately the effect on each phase of changing the proportion of negative and positive training examples.

A splitting criterion is a metric applied to an attribute that measures how homogeneous the induced subsets are, if a training set is partitioned based on the values of this attribute. Consider a discrete attribute A that has values $A = a_1$ through $A = a_m$ for some $m \geq 2$. In the two-class case, standard splitting criteria have the form

$$I(A) = \sum_{k=1}^{m} P(A = a_k) f(p_k, 1 - p_k)$$

where $p_k = P(j = 1|A = a_k)$ and all probabilities are frequencies in the training set to be split based on A. The function $f(p, 1 - p)$ measures the impurity or heterogeneity of each subset of training examples. All such functions are qualitatively similar, with a unique maximum at $p = 0.5$, and equal minima at $p = 0$ and $p = 1$.

Drummond and Holte [2000] have shown that for two-valued attributes the impurity function $2\sqrt{p(1-p)}$ suggested by Kearns and Mansour [1996] is invariant to changes in the proportion of different classes in the training data. We prove here a more general result that applies to all discrete-valued attributes and that shows that related impurity functions, including the Gini index [Breiman *et al.*, 1984], are not invariant to base rate changes.

Theorem 3: Suppose $f(p, 1 - p) = \alpha(p(1 - p))^\beta$ where $\alpha > 0$ and $\beta > 0$. For any collection of discrete-valued attributes, the attribute that minimizes $I(A)$ using f is the same regardless of changes in the base rate $P(j = 1)$ of the training set if $\beta = 0.5$, and not otherwise in general.

Proof: For any attribute A, by definition

$$I(A) = \alpha \sum_{k=1}^{m} P(a_k) P(j = 1|a_k)^\beta P(j = 0|a_k)^\beta$$

where a_1 through a_m are the possible values of A. So by Bayes' rule $I(A)/\alpha$ is

$$\sum_k P(a_k) (\frac{P(a_k|j = 1)P(j = 1)}{P(a_k)})^\beta (\frac{P(a_k|j = 0)P(j = 0)}{P(a_k)})^\beta.$$

Grouping the $P(a_k)$ factors for each k gives that $I(A)/\alpha$ is

$$\sum_k P(a_k)^{1-2\beta} (P(a_k|j = 1)P(j = 1)P(a_k|j = 0)P(j = 0))^\beta.$$

Now the base rate factors can be brought outside the sum, so $I(A)$ is $\alpha^{-1} P(j = 1)^\beta P(j = 0)^\beta$ times the sum

$$\sum_k P(a_k)^{1-2\beta} P(a_k|j = 1)^\beta P(a_k|j = 0)^\beta. \quad (3)$$

Because $\alpha^{-1} P(j = 1)^\beta P(j = 0)^\beta$ is constant for all attributes, the attribute A for which $I(A)$ is minimum is determined by the minimum of (3). If $2\beta = 1$ then (3) depends only on $P(a_k|j = 1)$ and $P(a_k|j = 0)$, which do not depend on the base rates. Otherwise, (3) is different for different base rates because

$$P(a_k) = P(a_k|j = 1)P(j = 1) + P(a_k|j = 0)P(j = 0)$$

unless the attribute A is independent of the class j, that is $P(a_k|j = 1) = P(a_k|j = 0)$ for $1 \leq k \leq m$. ■

The sum (3) has its maximum value 1 if A is independent of j. As desired, the sum is smaller otherwise, if A and j are correlated and hence splitting on A is reasonable.

Theorem 3 implies that changing the proportion of positive or negative examples in the training set has no effect on the structure of the tree if the decision tree growing method uses the $2\sqrt{p(1-p)}$ impurity criterion. If the algorithm uses a different criterion, such as the C4.5 entropy measure, the effect is usually small, because all impurity criteria are similar.

The experimental results of Drummond and Holte [2000] and Dietterich *et al.* [1996] show that the $2\sqrt{p(1-p)}$ criterion normally leads to somewhat smaller unpruned decision trees, sometimes leads to more accurate trees, and never leads to much less accurate trees. Therefore we can recommend its use, and we can conclude that regardless of the impurity criterion, applying Theorem 1 is not likely to have have much influence on the growing phase of decision tree learning.

4.3 Decision tree pruning

Standard methods for pruning decision trees are highly sensitive to the prevalence of different classes among training examples. If all classes except one are rare, then C4.5 often prunes the decision tree down to a single node that classifies all examples as members of the common class. Such a classifier is useless for decision-making if failing to recognize an example in a rare class is an expensive error.

Several papers have examined recently the issue of how to obtain good probability estimates from decision trees [Bradford *et al.*, 1998; Provost and Domingos, 2000; Zadrozny and Elkan, 2001b]. It is clear that it is necessary to use a smoothing method to adjust the probability estimates at each leaf of a decision tree. It is not so clear what pruning methods are best.

The experiments of Bauer and Kohavi [1999] suggest that no pruning is best when using a decision tree with probability

smoothing. The overall conclusion of Bradford *et al.* [1998] is that the best pruning is either no pruning or what they call "Laplace pruning." The idea of Laplace pruning is:

1. Do Laplace smoothing: If n training examples reach a node, of which k are positive, let the estimate at this node of $P(j = 1|x)$ be $(k + 1)/(n + 2)$.
2. Compute the expected loss at each node using the smoothed probability estimates, the cost matrix, and the training set.
3. If the expected loss at a node is less than the sum of the expected losses at its children, prune the children.

We can show intuitively that Laplace pruning is similar to no pruning. In the absence of probability smoothing, the expected loss at a node is always greater than or equal to the sum of the expected losses at its children. Equality holds only if the optimal predicted class at each child is the same as the optimal predicted class at the parent. Therefore, in the absence of smoothing, step (3) cannot change the meaning of a decision tree, i.e. the classes predicted by the tree, so Laplace pruning is equivalent to no pruning.

With probability smoothing, if the expected loss at a node is less than the sum of the expected losses at its children, the difference must be caused by smoothing, so without smoothing there would presumably be equality. So pruning the children is still only a simplification that leaves the meaning of the tree unchanged. Note that the effect of Laplace smoothing is small at internal tree nodes, because at these nodes typically $k \gg 1$ and $n \gg 2$.

In summary, growing a decision tree can be done in a cost-insensitive way. When using a decision tree to estimate probabilities, it is preferable to do no pruning. If costs are example-dependent, then decisions should be made using smoothed probability estimates and Equation (1). If costs are fixed, i.e. there is a single well-defined cost matrix, then each node in the unpruned decision tree can be labeled with the optimal predicted class for that leaf. If all the leaves under a certain node are labeled with the same class, then the subtree under that node can be eliminated. This simplification makes the tree smaller but does not change its predictions.

5 Conclusions

This paper has reviewed the basic concepts behind optimal learning and decision-making when different misclassification errors cause different losses. For the two-class case, we have shown rigorously how to increase or decrease the proportion of negative examples in a training set in order to make optimal cost-sensitive classification decisions using a classifier learned by a standard non cost-sensitive learning method. However, we have investigated the behavior of Bayesian and decision tree learning methods, and concluded that changing the balance of negative and positive training examples has little effect on learned classifiers. Accordingly, the recommended way of using one of these methods in a domain with differing misclassification costs is to learn a classifier from the training set as given, and then to use Equation (1) or Equation (2) directly, after smoothing probability estimates and/or adjusting the threshold of Equation (2) empirically if necessary.

References

[Bauer and Kohavi, 1999] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999.

[Bradford *et al.*, 1998] J. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C. Brodley. Pruning decision trees with misclassification costs. In *Proceedings of the European Conference on Machine Learning*, pages 131–136, 1998.

[Breiman *et al.*, 1984] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadswoth, Belmont, California, 1984.

[Dietterich *et al.*, 1996] T. G. Dietterich, M. Kearns, and Y. Mansour. Applying the weak learning framework to understand and improve C4.5. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 96–104. Morgan Kaufmann, 1996.

[Domingos and Pazzani, 1996] Pedro Domingos and Michael Pazzani. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 105–112. Morgan Kaufmann, 1996.

[Drummond and Holte, 2000] Chris Drummond and Robert C. Holte. Exploiting the cost (in)sensitivity of decision tree splitting criteria. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 239–246, 2000.

[Japkowicz, 2000] N. Japkowicz. The class imbalance problem: Significance and strategies. In *Proceedings of the International Conference on Artificial Intelligence*, Las Vegas, June 2000.

[Kearns and Mansour, 1996] M. Kearns and Y. Mansour. On the boosting ability of top-down decision tree learning algorithms. In *Proceedings of the Annual ACM Symposium on the Theory of Computing*, pages 459–468. ACM Press, 1996.

[Kubat and Matwin, 1997] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: One-sided sampling. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 179–186. Morgan Kaufmann, 1997.

[Margineantu, 2000] Dragos Margineantu. On class probability estimates and cost-sensitive evaluation of classifiers. In *Workshop Notes, Workshop on Cost-Sensitive Learning, International Conference on Machine Learning*, June 2000.

[Michie *et al.*, 1994] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

[Provost and Domingos, 2000] Foster Provost and Pedro Domingos. Well-trained PETs: Improving probability estimation trees. Technical Report CDER #00-04-IS, Stern School of Business, New York University, 2000.

[Weiss and Provost, 2001] Gary M. Weiss and Foster Provost. The effect of class distribution on classifier learning. Technical Report ML-TR 43, Department of Computer Science, Rutgers University, 2001.

[Zadrozny and Elkan, 2001a] Bianca Zadrozny and Charles Elkan. Learning and making decisions when costs and probabilities are both unknown. Technical Report CS2001-0664, Department of Computer Science and Engineering, University of California, San Diego, January 2001.

[Zadrozny and Elkan, 2001b] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001. To appear.

Mining Soft-Matching Rules from Textual Data

Un Yong Nahm and **Raymond J. Mooney**
Department of Computer Sciences,
University of Texas, Austin, TX 78712-1188
{pebronia,mooney}@cs.utexas.edu

Abstract

Text mining concerns the discovery of knowledge from unstructured textual data. One important task is the discovery of rules that relate specific words and phrases. Although existing methods for this task learn traditional logical rules, soft-matching methods that utilize word-frequency information generally work better for textual data. This paper presents a rule induction system, TEXTRISE, that allows for partial matching of text-valued features by combining rule-based and instance-based learning. We present initial experiments applying TEXTRISE to corpora of book descriptions and patent documents retrieved from the web and compare its results to those of traditional rule and instance based methods.

1 Introduction

Text mining, discovering knowledge from unstructured natural-language text, is an important data mining problem attracting increasing attention [Hearst, 1999; Feldman, 1999; Mladenić, 2000]. Existing methods for mining rules from text use a hard, logical criteria for matching rules [Feldman and Hirsh, 1996; Ahonen-Myka *et al.*, 1999]. However, for most text processing problems, a form of soft matching that utilizes word-frequency information typically gives superior results [Salton, 1989; Cohen, 1998; Yang, 1999]. Therefore, the induction of soft-matching rules from text is an important, under-studied problem.

We present a method, TEXTRISE, for learning soft-matching rules from text using a modification of the RISE algorithm [Domingos, 1996], a hybrid of rule-based and instance-based (nearest-neighbor) learning methods. Such a hybrid is good match for text mining since rule-induction provides simple, interpretable rules, while nearest-neighbor provides soft matching based on a specified similarity metric. Currently in TEXTRISE, we use the vector-space model from information retrieval (IR) to provide an appropriate similarity metric [Salton, 1989].

We present results on applying TEXTRISE to two text databases, one of book information extracted from an online bookstore, and another of patent applications available on the web. We evaluate the quality of the discovered rules on independent data by measuring the similarity of predicted text and actual text. By comparing results to the predictions made by nearest-neighbor and mined association rules, we demonstrate the advantage of mining soft-matching rules.

2 Background

2.1 Mining Rules from Text

Several researchers have applied traditional rule induction methods to discover relationships from textual data. FACT [Feldman and Hirsh, 1996] discovers rules from text using a well-known technique for *association rule mining*. For example, it discovered rules such as "Iraq ⇒ Iran", and "Kuwait and Bahrain ⇒ Saudi Arabia" from a corpus of Reuters news articles. Ahonen *et al.*(1998) also applied existing data mining techniques to discover *episode rules* from text. For example: "If "chemicals" and "processing" occurs within 2 consequent words, the word "storage" co-occurs within 3 words." is an episode rule discovered from a collection of Finnish legal documents.

In addition, decision tree methods such as C4.5 and C5.0, and rule learners such as FOIL, and RIPPER have been used to discover patterns from textual data [Nahm and Mooney, 2000b; Ghani *et al.*, 2000]. All of these existing methods discover rules requiring an exact match.

2.2 Mining Information Extracted from Text

Nahm and Mooney(2000a; 2000b) introduced an alternative framework for text mining based on the integration of *information extraction* (IE) and traditional data mining. IE is a form of shallow text understanding that locates specific pieces of data in natural-language text. Traditional data mining assumes that information is in the form of a relational database; unfortunately, in many applications, information is only available in the form of unstructured documents. IE addresses this problem by transforming a corpus of textual documents into a structured database. An IE module can extract data from raw text, and the resulting database can be processed by a traditional data mining component.

In this work, extracted textual data was mined using traditional rule induction systems such as C4.5rules [Quinlan, 1993] and RIPPER [Cohen, 1995]. Rules were induced for

predicting the text in each slot using the extracted information in all other slots. However, the heterogeneity of textual databases causes a problem: the same or similar objects are often referred to using different (but similar) textual strings. This issue becomes clear when we consider the Web, a vast and dynamic warehouse of text documents, as a potential target for text mining. Since the Web has no centralized moderator, it is highly heterogeneous, making it difficult to apply strict matching to text extracted from web documents [Cohen, 1998].

2.3 Information Retrieval Vector-Space Model

The vector-space model is typically used in IR to determine the similarity of two documents. In this model, a text is represented as a vector of real numbers, where each component corresponds to a word that appears in the set of all documents and the value is its frequency in the document. This is also known as a *bag of-words* (BOW) representation. The similarity of two documents x and y is the *cosine of the angle* between two vectors $\vec{x}$ and $\vec{y}$ representing x and y respectively, and calculated by the following formula:

$$Similarity(x, y) = \frac{\vec{x} \cdot \vec{y}}{\mid \vec{x} \mid \times \mid \vec{y} \mid} \quad (1)$$

where $\mid \vec{x} \mid$ and $\mid \vec{y} \mid$ are the norms of each document vector.

The TFIDF (Term Frequency, Inverse Document Frequency) weighting scheme [Salton, 1989] is used to assign higher weights to distinguished terms in a document. TFIDF makes two assumptions about the importance of a term. First, the more a term appears in the document, the more important it is (*term frequency*). Second, the more it appears through the entire collection of documents, the less important it is since it does not characterize the particular document well (*inverse document frequency*). In the TFIDF framework, the weight for term t_j in a document d_i, w_{ij} is defined as follows:

$$w_{ij} = tf_{ij} \times \log_2 \frac{N}{n} \quad (2)$$

where tf_{ij} is the frequency of term t_j in document d_i, N the total number of documents in collection, and n the number of documents where term t_j occurs at least once.

2.4 RISE: Learning Soft-Matching Rules

The RISE induction algorithm unifies rule-based and instance-based learning [Domingos, 1996]. Instead of requiring rules to match exactly, RISE makes predictions by selecting the closest matching rule according to a standard distance metric used by nearest-neighbor methods (a modified Euclidian distance). By generating generalized rules instead of remembering specific instances, and by using a similarity metric rather than exact matching to make predictions, it elegantly combines the properties of rule induction and instance-based learning.

Soft-matching rules are acquired using a specific-to-general (bottom-up) induction algorithm that starts with maximally specific rules for every example, and then repeatedly minimally generalizes each rule to cover the nearest example it does not already cover, unless this results in a decrease in the accuracy of the overall rule base on the training data. This process repeats until any additional generalization decreases accuracy. When classifying examples, the nearest rule is used to predict the class. A leave-one-out method is used to determine the performance of the rule base on the training data, since an example is always correctly classified by its corresponding initial maximally-specific rule. In extensive experiments, RISE was fairly consistently more accurate than alternative methods, including standard rule-based and instance-based algorithms. Training is also reasonably efficient computationally, requiring time $O(e^2a^2)$ where e is the number of examples, and a the number of attributes.

Book Description
Title : Harry Potter and the Goblet of Fire (Book 4)
Author : Joanna K. Rowling
Comments: This book was the best book I have ever read. If you are in for excitement this book is the one you want to read.
Subject: Fiction, Mystery, Magic, Children, School, Juvenile Fiction, Fantasy, Wizards

Representation
Author = {"joanna", "rowling"}
Title = {"harry", "potter", "goblet", "fire", "book"}
Comments = {"book", "book", "read", "excitement", "read"}
Subject = {"fiction", "mystery", "magic", "children", "school", "juvenile", "fiction", "fantasy", "wizards"}

Figure 1: An example of representation for a book document

3 The TEXTRISE Algorithm

RISE is not directly applicable to mining rules from extracted text because: 1) its similarity metric is not text-based and 2) it learns rules for classification rather than text prediction. TEXTRISE addresses both of these issues. We represent an IE-processed document as a list of bags of words (BOWs), one bag for each slot filler. We currently eliminate 524 commonly-occurring stop-words but do not perform stemming. Figure 1 shows an example for a book description and its BOW representation. Standard set-operations are extended to bags in the obvious way [Peterson, 1976]. A learned rule is represented as an antecedent that is a conjunction of BOWs for some subset of slots and a conclusion that is a predicted BOW for another slot (see Figure 4 for examples).

The standard TFIDF-weighted cosine metric is used to compute the similarity of two BOWs. The similarity of two examples (i.e. extracted documents) or rules is the average similarity of the BOWs in their corresponding slots. Bag intersection is used to compute the minimal generalization of two BOWs. The minimal generalization of two examples or rules is the minimal generalization of the BOWs in each of their corresponding slots. A rule is said to *cover* an example document if all of its antecedent BOWs are sub-bags of the example's corresponding BOWs. To extend the algorithm from classification to text prediction, we define a new measure for the accuracy of a rule set on an example set: $TextAccuracy(RS, ES)$ is the average cosine similarity of the predicted fillers for the examples in ES to the corresponding fillers predicted by a rule set RS. The algorithms for gen-

Inputs: $R = (A_1, A_2, ..., A_n, C_R)$ is a rule
$E = (E_1, E_2, ..., E_n, C_E)$ is an example.
A_i, E_i, C_R, and C_E are bags-of-words, possibly empty.
Output: R' is the generalized rule.
Function Most_Specific_Generalization (R, E)
For i := 1 to n do
  $A_i' := A_i \cap E_i$
$R' := (A_1', A_2', ..., A_n', C_R \cap C_E)$
Return R'.

Figure 2: Generalization of a rule to cover an example

Input: ES is the training set.
Output: RS is the rule set.
Function TextRISE (ES)
$RS := ES$.
Compute $TextAccuracy(RS, ES)$.
Repeat
  For each rule $R \in RS$,
    $\hat{E} := \underset{E \in ES'}{arg\,max}\, Similarity(E, R)$
      where ES' = {E': $E' \in ES$ and
        E' is not covered by R}
    R' := Most_Specific_Generalization(R, $\hat{E}$)
    RS' := RS with R replaced by R'
    If $TextAccuracy(RS', ES) \geq TextAccuracy(RS, ES)$
    Then $RS := RS'$
    If R' is identical to another rule in RS,
    Then delete R' from RS.
Until no increase in $TextAccuracy(RS, ES)$ is obtained.
Return RS.

Figure 3: The TEXTRISE rule-learning algorithm

eralizing a rule to cover an example and for learning rules are described in Figure 2 and Figure 3 respectively. The algorithm is a straightforward modification of RISE using the new similarity and predictive-accuracy metrics, and is used to induce soft-matching rules for predicting the filler of each slot given the values of all other slots. Our implementation makes use of the the BOW library [McCallum, 1996] for the bag-of-words text processing.

3.1 Interestingness Measures

The output of TEXTRISE is an unordered set of soft matching rules. Ranking rules based on an interestingness metric can help a human user focus attention on the most promising relationships. Several metrics for evaluating the "interestingness" or "goodness" of mined rules, such as *confidence* and *support*, have been proposed [Bayardo Jr. and Agrawal, 1999]. However, the traditional definitions for these metrics assume exact matches for conditions. Consequently, we modify these two common metrics for judging the goodness of soft-matching rules.

A *rule* consists of an antecedent and a consequent, and is denoted as $A \rightarrow C$ where A is equal to $A_1 \wedge A_2 \wedge ... \wedge A_i$. The *similarity-support* of an antecedent A, denoted as $simsup(A)$, is the number of examples in the data set that are soft-matched by A. In other words, $simsup(A)$ is the number of examples for which A is the closest rule in the rule base. The similarity-support of rule $A \rightarrow C$, denoted as $simsup(A \rightarrow C)$, is defined as the sum of similarities between C and the consequents of the examples soft-matched by A in the data set. In these definitions, we replace the traditional hard-matching constraints for a rule with weaker constraints determined relative to all the other rules in the rule base. Similarity-confidence of a rule $A \rightarrow C$, denoted by $simconf(A \rightarrow C)$, is computed as below.

$$simconf(A \rightarrow C) = \frac{simsup(A \rightarrow C)}{simsup(A)}$$

4 Evaluation

4.1 Data Sets

Two domains are employed in our evaluation of TEXTRISE: book data from `Amazon.com` and patent data downloaded from `Getthepatent.com`. We manually developed simple pattern-based IE systems or "wrappers" to automatically extract various labeled text-fields from the original HTML documents. The text extracted for each slot is then processed into a bag-of-words after removal of stop-words and remaining HTML commands.

The book data set is composed of 6 subsets, science fiction, literary fiction, mystery, romance, science, and children's books. 1,500 titles were randomly selected for each genre to make the total size of the book data set to be 9,000. A wrapper extracts 6 slots: `titles`, `authors`, `subject` terms, `synopses`, published `reviews`, and customer `comments`. Sample rules from this domain are given in Figure 4. Numbers associated to each word denotes the number of occurrences in the bag. The similarity-confidence and similarity-support values for each rule are also given.

3,000 patent documents were collected from dynamically generated web pages returned by a keyword search for "artificial intelligence". Four slots of `titles`, `abstracts`, `claims`, and `descriptions` are extracted for each patent. Sample rules are given in Figure 5.

4.2 Results

Unlike a standard rule learner that predicts the presence or absence of a specific slot value, TEXTRISE predicts a bag-of-words for each slot. Therefore, we evaluate the performance of TEXTRISE by measuring the average cosine similarity of the predicted slot values to the actual fillers for each slot. We compare the system to a standard nearest-neighbor method to show that TEXTRISE's compressed rule base is superior at predicting slot-values. In both methods, prediction is made by selecting the closest rule/example using only the text in the antecedent slots. We also tested nearest-neighbor without using information extraction to show the benefit of IE-based text mining. To clearly show IE's role, the only change made to nearest-neighbor was to treat the set of BOW's for the antecedent slots as a single, larger BOW.

The experiments were performed on the 9,000 book descriptions using ten-fold cross validation. Learning curves for predicting the `title` slot are shown in Figure 6. The graph shows 95% confidence intervals for each point. All the results on average similarities, precisions, and F-measures were statistically evaluated by a one-tailed, paired *t*-test. For

Rules from 2,500 Book Descriptions

title nancy(1), drew(1)
synopses nancy(1)
subject children(2), fiction(2), mystery(3), detective(3), juvenile(1), espionage(1)
→
author keene(1), carolyn(1)
$[49.71\%, 1.99]$

synopses role(1), protein(1), absorption(1), metabolism(4), vitamins(1), minerals(1)
reviews health(1)
subject science(1), human(1), physiology(1)
→
title nutrition(1)
$[27.87\%, 0.56]$

author beatrice(1), gormley(1)
synopses witness(1), ufo(1), landing(1)
subject science(1), fiction(2)
→
reviews aliens(1), ufo(1), book(2)
$[13.79\%, 0.34]$

title charlotte(1), perkins(1), gilman(1)
synopses work(1), utopias(1), herland(1), ourland(1)
reviews gilman(1), author(1)
subject literature(2), criticism(2), classics(1), women(1), literary(1)
→
comments utopia(1), feminist(1)
$[36.46\%, 0.73]$

title dance(1)
→
subject romance(2), fiction(2)
$[31.68\%, 0.98]$

Figure 4: Sample Rules from Book Descriptions

each training set size, two pairs of systems(TEXTRISE versus nearest-neighbor and nearest-neighbor versus nearest-neighbor without information extraction) were compared to determine if their differences were statistically significant ($p < 0.05$).

The results indicate that TEXTRISE does best, while nearest-neighbor without IE does worst. This shows TEXTRISE successfully summarizes the input data in the form of prediction rules. The rule-compression rate of TEXTRISE is about 80%, which means the number of rules TEXTRISE produces is 80% of the number of examples originally stored in the initial rule base. We conducted the same experiments for other slots, and found similar results except for predicting the `author` slot. In predicting the `author` slot, neither information extraction nor TEXTRISE improves performance over simple nearest-neighbor.

In addition to the textual similarity, we developed analogs for precision and recall. Precision and recall were defined as follows, where C is the correct BOW and P is the predicted one.

$$Precision = Similarity(C \cap P, P) \quad (3)$$

$$Recall = Similarity(C \cap P, C) \quad (4)$$

Rules from 3,000 AI-Related Patent Documents

abstract device(2), images(1)
claims invention(4), system(5)
description information(5), data(2), control(2), stored(2), point(1), user(3), application(2), flow(1), object(1), operation(1), software(1), storage(1)
→
title video(1)
$[9.44\%, 0.54]$

title automated(1)
claims device(4), based(1), determining(3), comprising(4), input(3), plurality(2), comprises(5), claim(7)
→
abstract apparatus(1)
$[7.25\%, 0.42]$

Figure 5: Sample Rules from Patent Documents

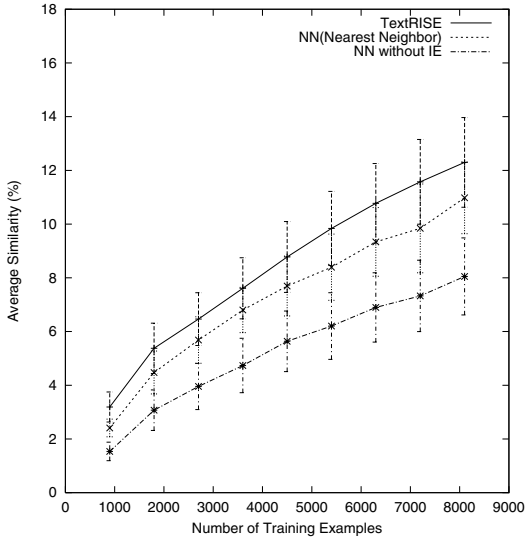


Figure 6: Average similarities for book data (`title`)

F-measure is defined as the harmonic mean for precision and recall as follows:

$$F-measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (5)$$

Learning curves for precision and F-measure are presented in Figure 7 and Figure 8. TEXTRISE provides higher precision, since the conclusions of many of its rules are smaller generalized BOWs, and overall F-measure is moderately increased.

To compare TEXTRISE with traditional rule mining methods, we generated association rules using the APRIORI algorithm [Agrawal and Srikant, 1994] and a publicly available implementation [Borgelt, 2000]. We treated each word in each slot as a separate item and generated associations between them. Among all the generated rules, those with words for the slot to be predicted are selected. For each test example, a prediction is made by building a BOW using the conclusions of all matching association rules. With the default

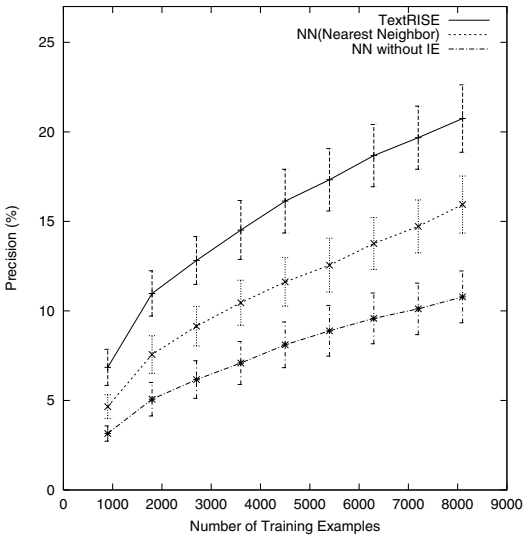


Figure 7: Precisions for book data (`title`)

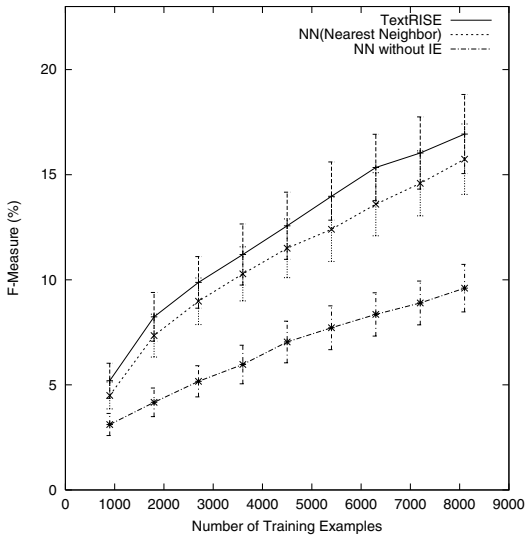


Figure 8: F-measures for book data (`title`)

parameter setting (minimum support of 10% and minimum confidence of 80%), the average similarity of predictions is almost 0%. We lowered the minimum support and the confidence until memory problems occurred on a SUN Ultra Sparc 1 (120MB). With the lowest minimum setting for support (3%) and confidence (30%), the average similarity remains very low: 0.0005% for 900 training examples and 0.0014% for 8,100 training examples. These results strongly suggest the usefulness of soft-matching rules in prediction tasks for textual data.

5 Related Research

Several previous systems mine rules from text [Feldman and Hirsh, 1996; Ahonen *et al.*, 1998]; however, they discover hard-matching rules and do not use automated information extraction. Ghani *et al.*(2000) applied several rule induction methods to a database of corporations automatically extracted from the Web. Interesting rules such as "Advertising agencies tend to be located in New York" were discovered; however, such learned rules must exactly match extracted text. WHIRL is a query processing system that combines traditional database and IR methods by introducing a "soft join" operation [Cohen, 1998]. WHIRL and TEXTRISE share a focus on soft-matching rules for text processing; however, rules in WHIRL must be written by the user while TEXTRISE tries to discover such rules automatically.

6 Future Work

A potential extension of the system is to generalize to a k-nearest-neighbor method that uses the k closest rules rather than just the single nearest rule. The predictions of these k rules could be combined by taking the average of the BOW vectors in their consequents. Likewise during learning, rules could be generalized to the k nearest uncovered examples using a similar averaging technique, possibly rounding values to maintain integer counts and simplify the resulting rules. Another potentially useful change to the generalization algorithm would be to use a semantic hierarchy such as WordNet [Fellbaum, 1998]. For example, the terms "thermodynamics" and "optics" could be generalized to "physics." Finally, for short extracted strings, string edit distance [Wagner and Fisher, 1974] might be a more useful measure of textual similarity than the cosine measure.

Better metrics for evaluating the interestingness of text-mined rules is clearly needed. One idea is to use a semantic network like WordNet to measure the semantic distance between the words in the antecedent and the consequent of a rule, preferring more "surprising" rules where this distance is larger. For example, this would allow ranking the rule "beer $\rightarrow$ diapers" above "beer $\rightarrow$ pretzels" since beer and pretzels are both food products and therefore closer in WordNet.

Although our preliminary results are encouraging, we are planning to evaluate the approach on other corpora such as a larger database of patents, grant abstracts from the National Science Foundation, or research papers gathered by CORA (`www.cora.whizbang.com`) or ResearchIndex (`citeseer.nj.nec.com`).

7 Conclusions

The problem of discovering knowledge in textual data is an exiting new area in data mining. Existing text-mining systems discover rules that require exactly matching substrings; however, due to variability and diversity in natural-language data, some form of soft matching based on textual similarity is needed. We have presented a system TEXTRISE that uses a hybrid of rule-based and instance-based learning methods to discover soft-matching rules from textual databases automatically constructed from document corpora via information extraction. With encouraging results of preliminary experiments, we showed how this approach can induce accurate

predictive rules despite the heterogeneity of automatically extracted textual databases.

Acknowledgements

This research was supported by the National Science Foundation under grant IRI-9704943.

References

[Agrawal and Srikant, 1994] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB-94)*, pages 487–499, Santiago, Chile, September 1994.

[Ahonen *et al.*, 1998] Helena Ahonen, Oskari Heinonen, Mika Klemettinen, and A. Inkeri Verkamo. Applying data mining techniques for descriptive phrase extraction in digital document collections. In *Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries*, pages 2–11, Santa Barbara, CA, April 1998.

[Ahonen-Myka *et al.*, 1999] Helena Ahonen-Myka, Oskari Heinonen, Mika Klemettinen, and A. Inkeri Verkamo. Finding co-occurring text phrases by combining sequence and frequent set discovery. In Ronen Feldman, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99) Workshop on Text Mining: Foundations, Techniques and Applications*, pages 1–9, Stockholm, Sweden, August 1999.

[Bayardo Jr. and Agrawal, 1999] Roberto J. Bayardo Jr. and Rakesh Agrawal. Mining the most interesting rules. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD-99)*, pages 145–154, San Diego, CA, August 1999.

[Borgelt, 2000] Christian Borgelt. Apriori version 2.6. http://fuzzy.cs.Uni-Magdeburg.de/~borgelt/, 2000.

[Cohen, 1995] William W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML-95)*, pages 115–123, San Francisco, CA, 1995.

[Cohen, 1998] William W. Cohen. Providing database-like access to the web using queries based on textual similarity. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD-98)*, pages 558–560, Seattle, WA, June 1998.

[Domingos, 1996] Pedro Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24:141–168, 1996.

[Feldman and Hirsh, 1996] Ronen Feldman and Haym Hirsh. Mining associations in text in the presence of background knowledge. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 343–346, Portland, OR, August 1996.

[Feldman, 1999] Ronen Feldman, editor. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99) Workshop on Text Mining: Foundations, Techniques and Applications*, Stockholm, Sweden, August 1999.

[Fellbaum, 1998] Christiane D. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

[Ghani *et al.*, 2000] Rayid Ghani, Rosie Jones, Dunja Mladenić, Kamal Nigam, and Sean Slattery. Data mining on symbolic knowledge extracted from the web. In Dunja Mladenić, editor, *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining (KDD-2000) Workshop on Text Mining*, pages 29–36, Boston, MA, August 2000.

[Hearst, 1999] Marti Hearst. Untangling text data mining. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, pages 3–10, College Park, MD, June 1999.

[McCallum, 1996] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/~mccallum/bow, 1996.

[Mladenić, 2000] Dunja Mladenić, editor. *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining (KDD-2000) Workshop on Text Mining*, Boston, MA, August 2000.

[Nahm and Mooney, 2000a] Un Yong Nahm and Raymond J. Mooney. A mutually beneficial integration of data mining and information extraction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 627–632, Austin, TX, July 2000.

[Nahm and Mooney, 2000b] Un Yong Nahm and Raymond J. Mooney. Using information extraction to aid the discovery of prediction rules from texts. In *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining (KDD-2000) Workshop on Text Mining*, pages 51–58, Boston, MA, August 2000.

[Peterson, 1976] James L. Peterson. Computation sequence sets. *Journal of Computer and System Sciences*, 13(1):1–24, August 1976.

[Quinlan, 1993] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo,CA, 1993.

[Salton, 1989] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[Wagner and Fisher, 1974] Robert A. Wagner and Michael J. Fisher. The string to string correction problem. *Journal of the Association for Computing Machinery*, 21:168–173, 1974.

[Yang, 1999] Yiming Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1/2):67–88, May 1999.