INDUCTION IN AN ABSTRACTION SPACE: A Form of Constructive Induction

George Drastal and Gabor Czako
Siemens Research and Technology Laboratories
755 College Rd
Princeton, NJ 08540

Stan Raatz
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

Abstract

We report on a learning system *MIRO* which performs supervised concept formation in an abstraction space. Given a domain theory, the method constructs this abstraction space by deduction over instances, and then performs induction in it rather than the initial space defined by instances alone. It is also possible to regard *MIRO* as a variant of constructive induction. The Vapnik-Chervonenkis model suggests that learning in an abstraction space can result in a substantial speedup, and we provide empirical studies which validate this proposition. We also show that learning in an abstraction space can reduce the number of false negative and false postive classifications because coincidental patterns are filtered by the deduction process. The method is able to extend an incomplete domain theory represented as attribute-value pairs with a set of rules that represent a disjunctive concept derived from a batch of training instances.

1. Introduction

The concept of abstraction has played an important and well-known role in artificial intelligence since the mid 1960's. The difficulty in making use of abstraction has always been the construction of an *explicit* mapping between the problem definition in the initial space and its definition in an abstraction space [1]. In this paper we report on the learning system *MIRO* which performs supervised concept formation in an abstraction space which is constructed by the process of deduction on rules composed of attribute-value pairs. The mapping is exactly the set of proof structures used in this construction. It then performs induction over a language defined by the process of deduction rather than the language defined by the instances, to yield a new characteristic concept description. The set of descriptors (or predicates) which are consequences of this deduction is viewed as the abstraction space, the set of descriptors used to describe the instances is viewed as the initial space, and the set of proof structures is viewed as the mapping between the two spaces.

This method can also be regarded as a form of construe tive induction in which "useful" patterns are encoded in the domain theory. However, we propose a strong form of deductive bias as a means of limiting the number of constructed descriptors. While the concept description formed is not justifiable in the sense of explanation-based generalization [3,12], the descriptors from which it is composed are justifiable." The method is able to extend an incomplete domain theory with a set of rules that represent a disjunctive concept derived from a batch of training instances. These inductively derived rules may be used in the same way as the original domain theory, when a new batch of instances is presented, and can support incremental refinement of the concept.

We present empirical studies which yield evidence for the following conjectures about learning in an abstraction space (as opposed to learning in the initial space). First, such learning can be more efficient because the abstraction space can be by construction more compact than is the initial space. Both the Vapnik-Chervonenkis characterization [16,17] of the complexity of learning from examples and our own studies suggest typical problems exist in which substantial speedup is expected. Second, learn ing in mi abstraction space can reduce the number of false negative and false postive classifications because coincidental patterns are filtered by the deduction process. We show evidence that a significant decrease in the number of misclassifications can be expected. The empirical studies mentioned are based on controlled, randomized, and exhaustive testing of many thousands of trials. This paper is a summary version, prepared for this conference, of reference [4] which is available by request.

2. Construction of an Abstraction Space

In essence, the basic idea presented here is a two stage process: first, construct an abstraction space, and then apply an induction method over this space. We will define the method for domain theories of rules composed of descriptor-values and an induction method similar to the Aq algorithm [9], but it is important to emphasize that the concept of induction in an abstraction space is

not restricted or dependent on these assumptions. We assume that training instances are given in the *initial* space $\mathcal{I} = \{0,1\}^n$ and a specific instance $\mathbf{x}$ is given by a set

$$\mathbf{x} = \{x_1, \ldots, x_k, \overline{x}_{k+1}, \ldots, \overline{x}_l\},$$

where for each $x_j \in \mathbf{x}$ $(1 \leq j \leq l)$, x_j is intepreted as descriptor x_j is present in instance $\mathbf{x}$ and $\overline{x}_j$ is interpreted as descriptor x_j is absent from instance $\mathbf{x}$. For a set $S = Pos \cup Neg$ of training instances, we say

$$L = \{x \mid x \in \mathbf{x}, \mathbf{x} \in S\}$$

is the *initial concept description language*. Let R_T be the target concept and let the rule space $\mathcal{R}$ in which it is represented be all k-term-DNF formulae over L. We refer to the learning problem in the *initial space* $\mathcal{I}$ as the problem of forming a characteristic concept description for R_T using the descriptors in L.

Let D be a domain theory of rules composed of descriptors as in

$$x \leftarrow x_1, \ldots, x_k, \overline{x}_{k+1}, \ldots, \overline{x}_n.$$

We assume that D does not contain a rule for deducing the "goal" concept R_T which characterizes the positive class. A *bottom-up* deduction procedure on the And/Or graph G_D induced by D can be defined by the following rules: truth is propagated from node n_1 to node n_2 along an Or-edge in G_D if n_1 is true, and truth is propagated from nodes $n_1, \ldots, n_k$ to node m along an And-edge in G_D if the nodes $n_1, \ldots, n_k$ are all true. Negation is represented by marking the edge from a negative literal as a *negative edge* and invoking a kind of negation as failure [2] rule: if a node labeled x cannot be marked true, then a node labeled $\overline{x}$ may be marked true, and that if a node labeled x can be marked true, then a node $\overline{x}$ cannot be marked true. Note that rules with negative consequents are not permitted in the domain theory. If instance $\mathbf{x}$ propagates truth to a descriptor x and the node labeled with x has no successor proven in the graph $G_D^{\mathbf{x}}$ for $\mathbf{x}$, we say x is a *maximally proven descriptor*, denoted $\mathbf{x} \vdash x$ *maximal*.

Let

$$L_{\mathcal{A}} = \{(x, \alpha) \mid \exists \mathbf{x} \in \{Pos \cup Neg\}, \mathbf{x} \vdash x \text{ maximal}, \\ \alpha \text{ a pointer to the node in } G_D^{\mathbf{x}} \text{ labeled } x\}.$$

The set $L_{\mathcal{A}}$ is *explicitly constructed* as a set of pairs consisting of a descriptor x and the *proof* that established x. We let $L_{\mathcal{A}}$ be the *abstract concept description language* and refer to the learning problem in an *abstraction space* $\mathcal{A}$ as the problem of forming a concept description for R_T using the descriptors in $L_{\mathcal{A}}$. The set of proofs or proof structures is the mapping from the initial to the abstraction space often mentioned in the literature [1].

Recently, the Vapnik-Chervonenkis (VC) dimension has come be recognized as a useful metric for estimating the number of training instances required to attain a nearly correct concept description (see [17] for details). For a finite rule space $\mathcal{R}$, the VC-dimension d is given by $d = log|\mathcal{R}|$ as developed in Rivest [16]. Our representation is k-term-DNF over n descriptors. There are 3^n possible 1-term-DNF rules and

$$|\mathcal{R}| = \binom{3^n}{k}$$

possible k-term-DNF formulae. Thus the VC-dimension is just

$$d = log|\mathcal{R}| = log\binom{3^n}{k},$$

which is approximately $k(nlog3 - loge)$ [4]. Let $k_{\mathcal{A}}$ stand for the value of k in the k-term-DNF representation of the abstraction space, $k_{\mathcal{I}}$ for the value of k in the initial space, $n_{\mathcal{A}}$ for the number of descriptors in the abstract language $L_{\mathcal{A}}$, and $n_{\mathcal{I}}$ for the number of descriptors in the initial language L. Thus if $k_{\mathcal{A}} < k_{\mathcal{I}}$ and $n_{\mathcal{A}} < n_{\mathcal{I}}$, the expected improvement in sample complexity when d is large is

$$O(\frac{k_{\mathcal{I}} n_{\mathcal{I}}}{k_{\mathcal{A}} n_{\mathcal{A}}}).$$

If $R_T \in \mathcal{R}_{\mathcal{I}}$ implies that $R_T \in \mathcal{R}_{\mathcal{A}}$, it is thus possible for the number of samples required to learn a nearly correct concept to be significantly smaller in the abstraction space than in the initial space. It remains to be seen if learning problems exist which exhibit this behavior. In section 4, we present empirical results which validate this claim.

3. Overview of Inductive Component

The inductive component reported here is directly applicable only to theories of rules composed of descriptor-value pairs, but illustrates topics that are applicable in other types of domain theories. The inductive stage is based on the one-sided variant [7] of the candidate elimination (or version space) algorithm [10]. Deduction provides a means of constructing a language L that is *justifiably biased*, in the sense that abstract descriptors are deductive consequents of the set of instance descriptors. This bias is exploited to its fullest if *only* maximal descriptors are admitted into L, and all instance descriptors are excluded. However, a maximally biased L may not be adequate for inducing a discriminant concept, even though the instance language is adequate. Since we cannot adjust the amount of bias in advance but prefer to use the strongest possible bias, we require an *unbiased inductor* capable of generating all discriminant concepts in L consistent with the training set. Candidate elimination is an efficient algorithm that meets this requirement. The algorithm is applied to a seed $\mathbf{x}_s \in Pos$ and to all negative instances to yield a set G of discriminant concept descriptions. Each element of G is a conjunction of descriptors drawn from a concept

description language and an element of *G* covers a subset of *Pos* including the seed. In general, this is a *partial concept description* that covers a subset of *Pos,* the positive training set. In case an empty *G* set is returned, one non-maximal descriptor is selected by a heuristic similar to Quinlan's decision tree heuristic [14] and added to *L*. This may be repeated until a nonempty *G* is returned, and a single conjunction *C* is selected that covers a subset of the positive instances. Selection from the *G* set depends on a heuristic measure of credibility [15] which balances the number of instances covered by a conjunction with an extra-evidential component that measures the amount of domain knowledge in the proof structure of each instance.

In order to obtain a characteristic partial description, each discriminant partial description is specialized by the addition of descriptors from an augmented language into the conjunction. The specialization entails a search guided by the same heuristic measure used to select an element from the *G* set. The instances covered by each partial characteristic description are then removed from the set *Pos.* A *complete concept description* is a disjunction of partial concept descriptions. Since we require a complete description, candidate elimination is applied repeatedly in order to construct a disjunction of terms. A new seed is used and one partial concept description is selected from the *G* set on each cycle. The algorithm terminates when either a complete description is found, or some positive instances remain which have been tried as seeds and cannot be discriminated from the set *Neg.* We summarize the method in pseudo-code as follows.

```
Construct abstract language L_A.
set characteristic concept description R ← ∅;
set L to all maximal descriptors in L_A;
do while Pos ≠ ∅
    Choose a seed x_s ∈ Pos;
    set G ← ∅;
    Invoke Candidate Elimination on x_s ∪ Neg,
            returning general version space G;
    do while G = ∅
        if All initial descriptors are already in L
            then Fail: indistinguishable instances
            else Select a descriptor and add to L;
        Restart Candidate Elimination with new L;
    end;
    Perform search from G, returning conjunction C;
    set R ← R ∨ C;
    set Pos ← Pos − {x|x ∈ Pos, x covered by C} ;
end;
```

4. Validation

A synthetic problem domain was invented that would exhibit sufficiently high dimensionality and disjunctiveness to represent difficult problems for a selective induction algorithm. The domain represents household cooking implements (e.g. soupbowls, forks) with an instance language of 28 observable, structural features. Many of the interesting classes that are represent able in this language share functional properties (e.g. insulated against heat) that are not included in the instance language. We invented a domain theory of 36 rules that represents such a functional property as the consequent of a rule, or chain of rules grounded in the instance language. These included four rules that could be interpreted as defining a class (glass, cup, plate, cooking vessel), but did not include any rules defining the target class for our learning experiments.

Training instances were created by pseudorandom mechanical generation of structure and features, in order to preclude the introduction of unconscious bias by the investigators. Several thousand rather bizarre objects resulted, and these were screened and mechanically sorted into training classes by using a classification rule base that was unknown to the learning program. A total of 78 usable instances resulted, of which 30 "spoons[11] were chosen for a pool of positive training instances. All remaining instances were used in the negative pool. An average instance of "spoon" has 14 features, and an average instance of any type has 15 features. The target concept is exactly represented by an 8-term-DNF expression in instance space.

A single learning trial consists of choosing *k, 2 < k <* 20, positive and *k* negative training instances from each pool, running *MIRO* to give a set of classification rules (i.e. a characteristic concept description), and then testing that concept against 10 positive and 10 negative instances chosen randomly from each pool excluding training instances used in that trial. By varying *k* from 2 through 20 by 2 we obtain a series. Each point in an error rate curve reported here is an average of 10 series. Figure 4.1 presents the false negative (solid line) and false positive (dotted line) error rates obtained in the initial feature space, using no domain theory. Here the false negative rate has not stabilized after 16 positive and negative instances, beyond which these trials often could not be completed due to exhausting LISP virtual memory. This explosion of memory use is clearly visible from figure 4.2, which plots growth of the version space G set during candidate elimination. The independent axes are the number of training instances (horizontal) and the number of negative instances eliminated (projection of axis perpendicular to the page). The dependent vertical axis shows the average size of *G*. We see clearly that induction is very under-constrained, owing to the highly disjunctive nature of the target concept when represented in the initial feature space.

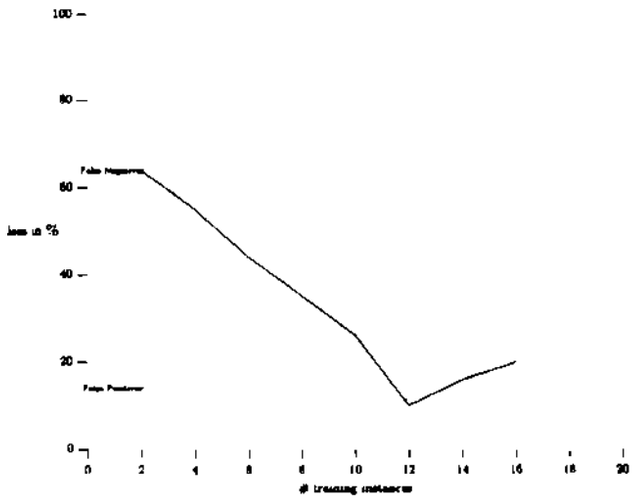

Figure 4.1 Initial space error rate

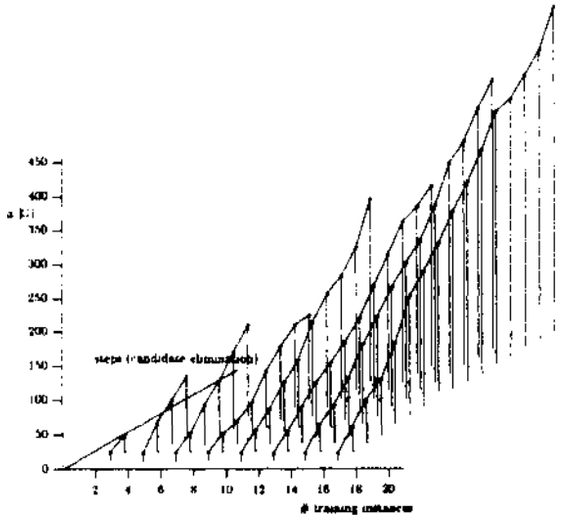

Figure 4.2 Initial space *G* set growth

The following experiment defines the performance baseline for *MIRO* in our synthetic domain, using a concept description language that was constructed as described in section 3 to exclude any initial descriptors. The resulting error rates in figure 4.3, now extended through 20 training instances, reach low plateaus far more rapidly. An average concept was observed to converge at 1-term 3-DNF around 4 training instances, and essentially no cases of G set collapse were observed. The effect of fully exploiting deductive bias is more visible in figure 4.4, which shows a slow growth in the G set as more training instances are used. These experiments suggest that an appropriate domain theory can reduce both the dimensionality and disjunctiveness of a learning problem, even though very little effort was made to engineer a "correct" or "complete" domain theorv.

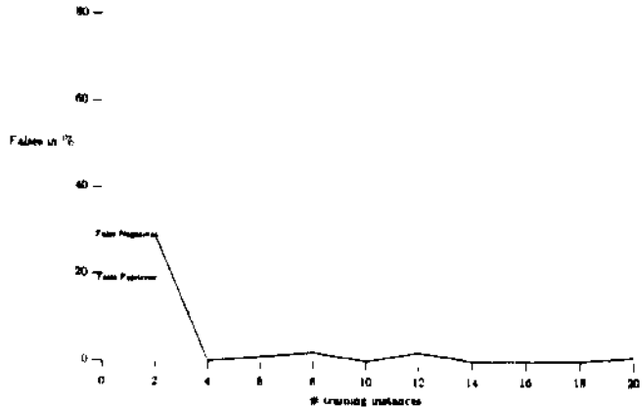

Figure 4.3 Abstraction space error rate

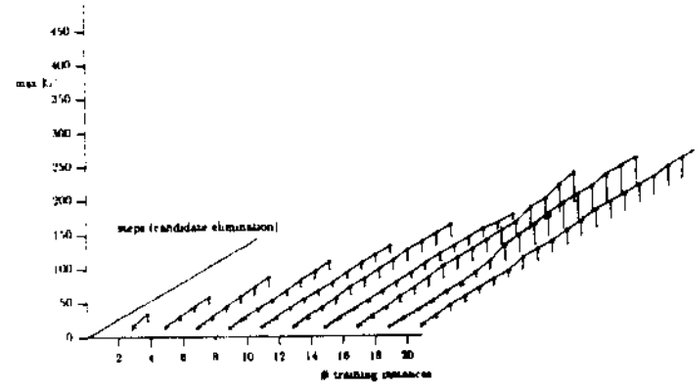

Figure 4.4 Abstraction space *G* set growth

5. Relationship to Other Work

The work reported in this paper is directly related to constructive induction as mentioned in the introduction, and to recent attempts to integrate explanation-based and inductive learning. However, there are important differences. Reference [4] also contains numerous other comparisons to work that is indirectly related, such as learning by analogy and various extensions to and uses of explanation-based learning, such as [11]. Lebowitz [8] develops a system *UNIMEM* which searches a database of voting records for empirical generalizations, and verifies these generalizations via a domain theory. He proposes an inductive

method that is used to control the search space for a version of a deductive method. This work can be considered the dual of the work reported in this paper, in the sense that we use a deductive technique to formulate a situation for induction. However, the work is very different in details, is presented by example and explanation, and neither includes an algorithm nor empirical studies. Pazzani, Dyer and Flowers [13] describe a system *OCCAM* in which causal theories are preferred to correlational or inductive information in forming generalizations, which are subsequently used to suggest additional causal and intentional relationships. The work is also not directly related, since *OCCAM* does not have a uniform language that is used in both the deductive and inductive stages, that is, it is not actually a form of constructive induction. Flann and Dietterich [6] propose a general learning architecture that uses a multiple representation strategy: the system translates task training examples into a ''natural" representation for induction and then translates the learned concept back into the appropriate task representation. The paradigm is illustrated by the system Wyl which learns concepts in board games such as checkers and chess. It is similar to the work reported in this paper because in Wyl all possible board positions consistent with a functional description of a concept in logic are constructively generated, but the architecture upon which Wyl is based appears to be more general.

6. Conclusions

We have presented evidence that induction defined over an abstraction space constructed via the process of deduction can result in substantial speed up for some induction problems, and that it is also possible for the number of false negative and false positive classifications to be reduced. In addition, it can be shown [5] that even with injection of 25% of attribute noise into the training set, the method presented here is able to construct a "corrected" characteristic concept description, and that with certain "ad hoc" assumptions, the method can accept domain theories in the Horn subset of first-order logic.

7. References

[1] Amarel, S., On Representations of Problems of Reasoning About Actions, in *Machine Intelligence* 3 (Miehie, D., ed.), Edinburgh University Press, Edinburgh (1968), 131-171. Also, in *Readings in Artificial Intelligence,* Tioga Press, Palo Alto, CA, 1981.

[2] Clark, K., Negation as Failure, in Logic and Databases, Gallaire and Minker (eds.), Plenum Press, NY (1978), 293-322.

[3] DeJong, G. and Mooney, R., Explanation-based Learning: An Alternate View, *Machine Learning* 1:2, 145-176, 1986.

[4] Drastal, G. and Raatz, S., Empirical Results on Induction in an Abstraction Space, Department of Computer Science DCS-TR-248 Rutgers University, November 1988.

[5] Drastal, G., Raatz, S., and Meunier, R., Error Correction in Constructive Induction, submitted to this conference.

[6] Flann. N. and Dietterich, T., Selecting Appropriate Representations for Learning from Examples, *AAAI-86,* Philadelphia, PA, 460-466.

[7] Haussler, D., Bias, version spaces and Valiant's learning framework, Proceedings *4th* International Workshop on Machine Learning, 324-336, Irvine, CA, 1987.

[8] Lebowitz, M., Integrated Learning: Controlling Explanation, *Cognitive Science* 10 (1986), 219-240.

[9] Michalski, R., AQVAL-1, Computer Implementation of a Variable valued Logic System VL I and Examples of its Application to Pattern Recognition, in Proceedings 1"' International Joint Conference on Pattern Recognition, Washington DC (1973), 3-17.

10] Mitchell, T., Version spaces: An approach to concept learning. Ph.D. thesis, Department of Electrical Engineering, Stanford University, 1978.

11] Mitchell, T., Toward Combining Empirical and Analytic Methods for Learning Heuristics, in *Human and Artificial Intelligence,* Elithorn and Banerji (eds.), North-Holland Publishing Co., Amsterdam.

12] Mitchell, T., Keller, R., and Kedar-Cabelli, S., Explanation-based Generalization: A Unifying View. *Machine Learning* 1:1, 47-80, 1986.

13] Pazzani, M., Dyer, M., and Flowers, M., The Role of Prior Theories in Generalization, AAAI-86, Philadelphia, 545-550.

14] Quinlan, J., Induction of decision trees, *Machine Learning* 1:1, 81-106, 1986.

15] Rendell, L., A general framework for induction and a study of selective induction, *Machine Learning* 1:2, 177 226, 1986.

16] Rivest, R., Notes on Machine Learning, unpublished draft.

17] Vapnik, V., and Chervonenkis, A., On the uniform convergence of relative frequencies of events to their probabilities, *Theory of Probability and its Ap])lications* 16, 264-280, 1971.