Sample Efficient On-Line Learning of Optimal Dialogue Policies with Kalman Temporal Differences

Olivier Pietquin[†,‡] and **Matthieu Geist**[†] and **Senthilkumar Chandramohan**[†]
[†]SUPELEC - IMS Research Group
[‡]UMI 2958 (GeorgiaTech - CNRS)
2 rue Edouard Belin, 57070 Metz - France
email : firstname.lastname@supelec.fr*

Abstract

Designing dialog policies for voice-enabled interfaces is a tailoring job that is most often left to natural language processing experts. This job is generally redone for every new dialog task because cross-domain transfer is not possible. For this reason, machine learning methods for dialog policy optimization have been investigated during the last 15 years. Especially, reinforcement learning (RL) is now part of the state of the art in this domain. Standard RL methods require to test more or less random changes in the policy on users to assess them as improvements or degradations. This is called *on policy* learning. Nevertheless, it can result in system behaviors that are not acceptable by users. Learning algorithms should ideally infer an optimal strategy by observing interactions generated by a non-optimal but acceptable strategy, that is learning *off-policy*. In this contribution, a *sample-efficient*, *online* and *off-policy* reinforcement learning algorithm is proposed to learn an optimal policy from few hundreds of dialogues generated with a very simple handcrafted policy.

1 Introduction

After 60 years of research, speech and language processing techniques have come to such a level of maturity that voice-enabled interfaces are now industrialized and have the potential of creating billions of profits. Yet, the design of such interfaces is far from being simple and requires expert skills in speech and language technologies, ergonomics, low level programming, *etc*. Indeed, it is not enough to put speech recognition and synthesis systems together to build a natural interface but the *management* of the interaction is responsible for its efficiency and naturalness. Dialog management is highly task dependent and it has to be designed again for each new application. Hiring or training personals with all the required skills is very difficult as well as expensive and this probably slows down the development of speech-based interfaces for the general public.

*The work presented here has been done during the CLASSiC project (Grant No. 216594, www.classic-project.org) funded by the European Commission's 7th Framework Programme (FP7).

Because of this, research in the domain of Spoken Dialogue Systems (SDS) have experienced an increasing growth during the last decade. Machine learning, and especially Reinforcement Learning (RL) [Sutton and Barto, 1998], for optimal dialog management policy learning is now part of the state of the art [Singh *et al.*, 1999; Levin *et al.*, 2000; Pietquin and Dutoit, 2006; Williams and Young, 2007]. RL is a machine learning method for optimizing sequential decision making by maximizing a cumulative of local rewards obtained through actual interactions as described in Section 2. Dialogue management can indeed be seen as a sequential decision making problem where a dialog manager has to select which information to ask or transmit to a human user according to the context of the dialogue. Yet, standard RL algorithms are very data demanding and dialog corpora are very hard to collect and annotate. To alleviate this data sparsity problem, dialogue simulation based on user modeling is most often used to artificially expand training datasets [Schatzmann *et al.*, 2006; Pietquin and Dutoit, 2006; Levin *et al.*, 2000]. However, the learnt strategies are sensible to the quality of the user model which is very difficult to assess [Schatzmann *et al.*, 2005].

An alternative to this bootstrapping method would be to use generalization frameworks adapted to RL such as approximate dynamic programming [Gordon, 1995; Lagoudakis and Parr, 2003] or online value function approximation [Sutton and Barto, 1998]. Very few attempts to do so in the framework of dialogue management can be found in the literature. In [Henderson *et al.*, 2008], the authors use the SARSA(λ) algorithm [Sutton and Barto, 1998] with linear function approximation which is known to be sample inefficient. In [Li *et al.*, 2009], LSPI [Lagoudakis and Parr, 2003] is used with feature selection and linear function approximation. Recently, Fitted Value Iteration (FVI) [Gordon, 1995] have also been applied to dialogue management [Chandramohan *et al.*, 2010; Pietquin *et al.*, 2011]. All these studies report *batch* learning of dialog policies from fixed sets of data and thus learn in an *off-policy* manner, meaning that they learn an optimal policy from observations generated with another policy (which is mandatory for learning from fixed sets of data). Online learning using Gaussian Processes [Gasic *et al.*, 2010] or Natural Actor Critic [Jurcicek *et al.*, 2010] has also been proposed. These works report the use of *online* and *on-policy* algorithms which requires permanently changing the policy to be learnt

(an issue known as the dilemma between exploration and exploitation). These changes to the policy made during learning are visible to the user which may cause problems in real applications at the early stage of learning where the changes in the policy can lead to very bad behaviors of the dialogue manager. Thus, user simulation is still required.

This contribution proposes a *sample-efficient*, *online* and *off-policy* learning algorithm for dialogue management policy optimization, namely the Kalman Temporal Differences (KTD) algorithm [Geist and Pietquin, 2010a][1].This algorithm meets many requirements of this specific task. First, off-policy learning is mandatory because collecting data requires real interactions with users which needs a fairly good initial dialogue policy. So the optimal policy has to be learnt from observations generated by another policy producing an acceptable behavior (even if suboptimal) by the users. Second, sample-efficient online learning can react rapidly to changes in the average behavior of users. Such changes can occur because of the users getting used to interacting with the system. It should of course use as few samples as possible because of the cost of data collections. KTD is compared to other offline (FVI and LSPI) and online (Q-learning) off-policy methods. Experimental results show that KTD learns a policy that compares positively to those learnt by other algorithms with only few tens of dialogue samples while several thousands of dialogues are required by the others to reach convergence.

2 Dialogue Management and Reinforcement Learning

In the general RL paradigm [Sutton and Barto, 1998], an agent learns to control optimally a dynamic system through actual interactions with this system. At each time step i, the system is in a given state s_i and the agent chooses an action a_i according to its current strategy (or *policy*) π and to the current state s_i. Following its own dynamics, the system is then driven (stochastically) to a new state s_{i+1}, and the agent receives a reward r_i which is a local hint on the quality of the strategy. The goal of RL is to learn a policy maximizing the expected cumulative reward on the long run (and not the immediate rewards).

2.1 Markov Decision Process

Formally, the problem can be cast into the Markov Decision Processes (MDP) paradigm [Bellman, 1957]. An MDP is a tuple $\{S, A, P, R, \gamma\}$ where S is the state space, A is the action space, $P \in \mathcal{P}(S)^{S\times A}$ is the set of *Markovian* transition probabilities[2], $R \in \mathbb{R}^{S\times A\times S}$ is the reward function and γ is a discount factor weighting long-term rewards. A policy is a mapping from states to actions: $\pi \in A^S$. The quality of a policy π is quantified thanks to the so-called value function $V^\pi \in \mathbb{R}^S$ which associates to each state the expected discounted cumulative reward (expectation being done according to trajectories) obtained by starting in this state and then following the policy π:

$$V^\pi(s) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, \pi] \quad (1)$$

The state-action value (or Q-) function ($Q^\pi : \mathbb{R}^{S\times A}$) adds a degree of freedom for the choice of the first performed action:

$$Q^\pi(s,a) = E[\sum_{k=0}^{\infty} \gamma^k r_k | s_0 = s, a_0 = a\pi] \quad (2)$$

An optimal policy π^* maximizes the value of each state:

$$\pi^* \in \operatorname*{argmax}_\pi V^\pi (\text{or } \in \operatorname*{argmax}_\pi Q^\pi) \quad (3)$$

The optimal policy is *greedy* respectively to the optimal Q-function Q^*: $\pi^*(s) = \operatorname{argmax}_{a\in A} Q^*(s,a)$. Finding the optimal policy therefore reduces to finding the optimal Q-function. Because of the Markov property of transition probabilities, the optimal Q-function can be expressed by the Bellman optimality equation :

$$Q^*(s,a) = E_{s'|s,a}[R(s,a,s') + \gamma \max_{b\in A} Q^*(s',b)] \quad (4)$$

Most often, the state-action space is too large to obtain an exact representation of the Q^* function. In such a case, it is convenient to adopt a parametric representation. Let $\theta \in \mathbb{R}^p$ represent the parameter vector and $\hat{Q}_\theta$ the approximate Q-function belonging to the hypothesis space $\mathcal{H}$ (the functional space spanned by parameters). The goal is therefore to compute a good approximation $\hat{Q}_\theta$ of Q^*. For example, considering a linear parameterization:

$$\mathcal{H} = \left\{ \hat{Q}_\theta : \hat{Q}_\theta(s) = \sum_{j=0}^{p} \theta_j \phi_j(s) \right\} \quad (5)$$

where $\phi_j(s)$ are *features* or *basis functions*, for instance Gaussian functions. In this case, the Q-function is approximated by a weighted sum of such basis functions, the parameter vector to be found being the weights θ.

2.2 Dialogue Management as an MDP

Dialogue management can be seen as a sequential decision making problem where a dialogue manager has to select which information should be asked or provided to the user when in a given dialogue context. It can thus be cast into the MDP framework. The set A of *actions* a dialog manager can select is defined by the so called *dialog acts* it can perform. There can be different types of dialog acts such as: greeting the user, asking for a piece of information, providing a piece of information, asking for confirmation about a piece of information, closing the dialog *etc.* The *state* of a dialog is usually represented efficiently by the Information State paradigm [Larsson and Traum, 2000]. In this paradigm, the dialogue state contains a compact representation of the history of the dialogue in terms of dialog acts and its user responses. It summarizes the information exchanged between the user and the system until the considered state is reached. It is a representation of the context. A dialogue management

[1]The previous work by the authors focused on theoretical aspects of KTD and artificial benchmarks.

[2]Notation $f \in A^B$ is equivalent to $f : B \to A$.

strategy or policy π is therefore a mapping between dialogue states and dialogue acts. According to the MDP framework, a reward function has to be defined. The immediate reward is often modeled as the contribution of each action to the user's satisfaction [Singh *et al.*, 1999]. This is a subjective reward which is usually approximated by a linear combination of objective measures (dialogue duration, number of ASR errors, task completion *etc.*). Weights of this linear combination can be computed from empirical data [Walker *et al.*, 1997].

3 Kalman Temporal Differences

There exists many RL algorithms in the literature [Sutton and Barto, 1998] aiming at estimating the optimal Q-function for a given MDP (see eq. (1)). The Kalman Temporal Differences (KTD) framework [Geist and Pietquin, 2010a] has been recently introduced as an alternative way to estimate this Q-function. The basic idea behind this work is to cast value function approximation as a filtering problem, and to solve it using Kalman filtering [Kalman, 1960] (as well as variations). Like Kalman filtering extensions, it allows handling nonlinearities (nonlinear parameterizations but also the max operator of eq. (4)), handling non-stationarity which can be helpful to adapt to changes in the average behavior of users but also active sampling of the state-action space [Geist and Pietquin, 2010b]. This approach is also very sample efficient and therefore allows learning good policies from very few samples. Moreover, it can be used in an off-policy setting, making the algorithm able to learn optimal policies while simply observing sub-optimal policies. That is the aspect investigated here. The KTD-Q algorithm is the KTD's specialization devoted to learning the optimal state-action value function, and it is briefly presented here.

3.1 Q-function approximation as a filtering problem

A filtering problem consists in estimating hidden quantities X from related observations Y. For instance, estimating the position of an object from noisy observations (case of the radar). In [Kalman, 1960], the author proposes the well-know Kalman filter solution which is a statistical sequential solution. The quantities X and Y are modeled as random variables, the searched solution is the conditional expectation $\hat{X}_{i|i} = E[X|Y_1, \ldots, Y_i]$ and the algorithm is based on a so-called state-space formulation (not to be confounded with the state space of the MDP). This formulation specifies (*via* 2 equations) how the hidden quantities evolve (see eq. (6)), and how they are linked to observations (see eq. (7)).

$$X_{i+1} = f_i(X_i, v_i), \tag{6}$$

$$Y_i = g_i(X_i, w_i). \tag{7}$$

Given these equations, the Kalman filter proposes a linear update rule providing the best linear estimate of the conditional expectation $\hat{X}_{i|i} = E[X|Y_1, \ldots, Y_i]$:

$$\hat{X}_{i|i} = \hat{X}_{i|i-1} + K_i(Y_i - \hat{Y}_{i|i-1}) \tag{8}$$

where K_i is the so-called Kalman gain (which is computed according to some statistics not developed here) and $\hat{Y}_{i|i-1}$ is the prediction of the observation given the current estimate of $\hat{X}_{i|i-1}$ and eq. (7). In the KTD framework, the hidden quantities to be estimated are the Q-function parameters ($X = \theta$), and the observations are the rewards ($Y = r_i$) (as well as related transitions from states to states given actions: $\{s, a, s'\}$). Optimal parameters can evolve with time, meaning that the optimal policy is not fixed (so is the associated Q-function). It is the case if the system is not stationary for example, which happens in dialogue systems because the user population evolves for instance. Yet, this evolution is not known (the function f of eq. (6) is not known) and, adopting the Occam's razor principle, a random walk evolution model is assumed for parameters ($\theta_i = \theta_{i-1} + v_i$ with v_i being a centered white evolution noise). Rewards are linked to parameters through the (sampled) Bellman optimality equation (see eq. (4)) which provides the g_i function of eq. (7). A centered and white observation noise n_i is added because the estimated value Q_θ doesn't necessary live in the functional space $\mathcal{H}$ (inductive bias). This provides the following state-space formulation in the Kalman filtering paradigm:

$$\begin{cases} \theta_i = \theta_{i-1} + v_i \\ r_i = \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \max_{a \in A} \hat{Q}_{\theta_i}(s_{i+1}, a) + n_i \end{cases} \tag{9}$$

According to the Kalman Filtering theory, the KTD-Q algorithm provides among all linear estimators the one minimizing the expectation (remember that parameters θ_i are modeled as random variables) of the mean-squared error conditioned on past observed rewards: $J_i(\theta) = E[\|\theta_i - \theta\|^2 | r_{1:i}]$.

The full algorithm is not given here for the seek of brevity (all details can be found in [Geist and Pietquin, 2010a]), however the general update rule is provided. Parameters θ_i but also an associated variance P_i are maintained. For each new transition, some statistics P_{r_i} (scalar) and $P_{\theta r_i}$ (vector) as well as the predicted reward $\hat{r}_i$ (basically, this predicted reward is computed using the observation equation, and $r_i - \hat{r}_i$, the difference between the observed reward and its prediction, is actually a temporal difference error) are computed (using notably θ_{i-1} and P_{i-1}), and parameters (as well as the variance matrix) are updated using according to the Kalman gain K_i:

$$\begin{cases} K_i = P_{\theta r_i} P_{r_i}^{-1} \\ \theta_i = \theta_{i-1} + K_i(r_i - \hat{r}_i) \\ P_i = P_{i-1} - K_i P_{r_i} K_i^T \end{cases} \tag{10}$$

4 Application to a dialog management problem

The considered system to control is a form-filling and task-oriented spoken dialog system. The problem is a tourism information task, similarly to the one in [Lemon *et al.*, 2006]. Its goal is to provide information about restaurants in a city based on specific user preferences. There are three slots in this dialog problem, namely the location of the restaurant in the city, the cuisine of the restaurant and its price-range. Given past interactions with the user, the agent has to ask questions so as to propose the best choice according to the user preferences.

4.1 Related MDP

The corresponding MDP's state has 3 continuous components ranging from 0 to 1, each representing the averaging of filling and confirmation confidence scores (provided by the speech recognition system) of the respective slots. There are 13 possible actions: ask for a slot (3 actions), explicit confirmation of a slot (3 actions), implicit confirmation of a slot and ask for another slot (6 actions) and close the dialog by proposing a restaurant (1 action). The corresponding reward is always 0, except when it closes the dialog. In this case, the agent is rewarded 25 per correct slot filling, -75 per incorrect slot filling and -300 per empty slot. The discount factor is set to $\gamma = 0.95$ and it thus favors shorter interactions (see eq. (2)).

4.2 Experimental setup

Even if the ultimate goal is to implement RL on a real dialog management problem, a user simulation technique was used here to generate data. This way, the sensibility of learning can be analyzed, and obtained policies can be tested sufficiently often to obtain meaningful statistics. The user simulator was plugged to the DIPPER dialogue management system [Lemon *et al.*, 2006] to generate dialogue samples. To generate data, the dialogue manager strategy was based on a simple hand-coded policy (which aims at filling each slot by explicitly asking before closing dialog) combined with random action selection. With a probability ϵ, the system acts randomly, and with a probability $1 - \epsilon$, the system acts according to the specified policy. Random actions are actually chosen among "reasonable actions" and a set of hand-crafted rules prevents the dialogue manager to confirm a slot while it has never been asked before, for instance. This ensures that the state-action space is sufficiently explored (even if considered algorithms are off-policy, they cannot learn what to do in situations which are not similar to any seen dialogue turn).

KTD-Q is compared to 3 other algorithms, namely fitted-Q [Gordon, 1995], LSPI [Lagoudakis and Parr, 2003] and Q-learning [Sutton and Barto, 1998]. All algorithms are off-policy. Fitted-Q and LSPI are sample-efficient batch algorithms. Q-learning is an online algorithm (the off-policy variation of SARSA, the mostly used algorithm in the SDS literature). KTD is a sample-efficient online algorithm.

4.3 Reinforcement Learning setup

The Q-function is represented using one Radial Basis Function (RBF) network per action. Each RBF network has three equi-spaced Gaussian functions per dimension, each one with a standard deviation of $\sigma = \frac{1}{3}$ (state variables ranging from 0 to 1). Therefore, there are 351 (*i.e.*, $3^3 \times 13$) parameters. The corresponding feature vector is zero everywhere, except in the 9 components corresponding to the considered actions which are of the form $\exp(-\frac{\|s-s_i\|^2}{2\sigma^2})$, s_i being the i^{th} RBF center. For all the algorithms, the initial parameter vector θ_0 is set to 0. For LSPI and fitted-Q, no parameters have to be tuned (which is an advantage of these approaches). For Q-learning, one has to choose the learning rate α. Here a learning rate of 0.2 is chosen, divided by two each 10^4 interactions (similarly to what is done in [Lemon *et al.*, 2006] for SARSA). For KTD-Q, a prior variance over parameters (matrix P_0) and the variance of the observation noise have to be chosen. It is less usual than choosing a learning rate, but not really more difficult (see [Geist and Pietquin, 2010a] for details about parameters tuning for KTD in general).

4.4 Results

Obtained results are presented on figures 1 (abscissa in linear scale) and 2 (abscissa in logarithmic scale). These curves show the mean discounted cumulative rewards and associated standard deviations of each learnt policy in addition to the hand-coded policy. The hand-coded policy is not provided as a baseline but only to show that the learnt policies (with all the algorithms) were obtained by observing dialogues generated with this quite poor policy (there is no need of a good initial policy). The abscissa represents transitions (or dialog turns) and not dialogs. In average, a dialog obtained with an optimal policy is composed of 4 to 8 turns. To compute these statistics, each algorithm was trained 8 times for 5.10^4 interactions on different simulations (for each training phase, all algorithms used the same transitions), and each obtained policy was tested 50 times. Therefore, each point is an average of 400 runs. Notice that we stopped each testing dialog if not finished after 100 turns. The associated reward is close to 0. Therefore, a value of 0 means that the agent did not learn to stop the dialog, and it is therefore normal that the associated standard deviation is low. Fitted-Q and LSPI curves start only at 5.10^3 samples: these algorithms require inverting some matrices which are too badly conditioned below a few thousands of samples (knowing that we have a few hundreds of parameters)[3].

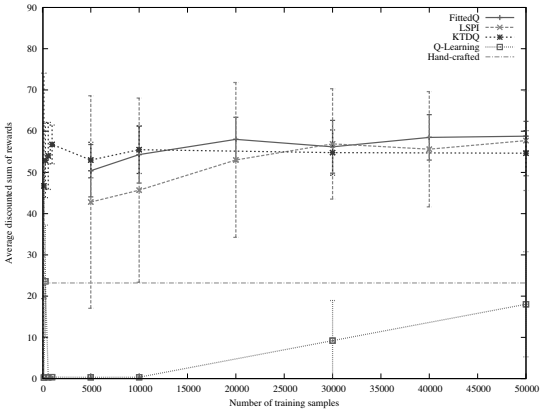

Figure 1: Results, linear scale.

According to figure 1, Q-learning learns very slowly, it does not even learn a policy as good as the hand-coded one after 5.10^4 interactions. This is compliant with the literature [Lemon *et al.*, 2006]. KTD-Q learns near optimal policies in very few samples. They are a little bit underperforming compared to those learnt by LSPI or fitted-Q. However,

[3] Although the convergence problem can be solved by regularizing the matrices to be inverted, our experiments showed that regularized LSPI resulted in unstable policies.

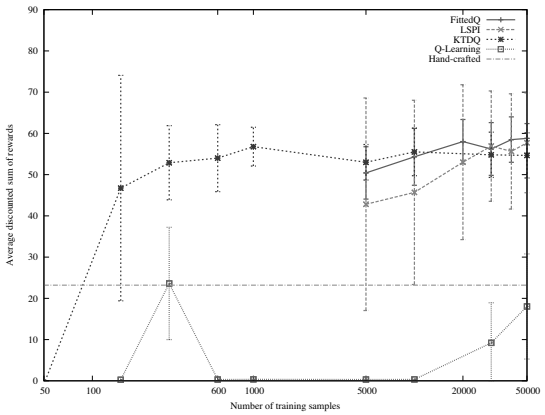

Figure 2: Results, log. scale.

recall that KTD-Q is an online algorithm. Therefore, it process each sample only one-time, contrary to fitted-Q or LSPI which process each transition many times in the regression process. One can notice that KTD-Q learns good policies very fast, that is why we provide on figure 2 the same curves with a logarithmic scale for the x-axis.

This figure shows that after 300 hundred transitions, Q-learning learns a not so bad policy (approximately equivalent to the hand-crafted policy). However, it looses this performance quite quickly, learning is not stable at this stage (according to experiments not reported here, tuning the learning rate does not help here).On the other hand, the KTD-Q algorithm learns good policies (performing equally to LSPI after convergence) after only a few hundreds of transitions, and in a stable manner (variance is decreasing). Another interesting aspect is its ability to provide good policies when there is not enough sample to even run fitted-Q or LSPI (they do not converge). So, 500 turns (about 100 dialogues) are sufficient to learn a near-optimal policy while standard Q-learning would require 1000 times more. This amount of dialogs can be easily collected in real applications to learn optimal policies with KTD-Q. Also, the variance is much lower with KTD after 5000 turns than for LSPI meaning that results are more reproducible.

5 Discussion

Results presented in the previous section are very promising. The proposed online algorithm could learn high performance policies from few tens of dialogue turns collected with a very simple strategy (its performance is quite bad when compared to the learnt ones) and without having to change the policy while interacting with users during the learning phase. One should remember that the learning curves are drawn with respect to dialogue turns and not dialogues. In average, a complete learnt dialogue was about 4 to 5 turns long (it is shorter than the optimal control, but it does not always satisfies the user). Since the Fitted-Q and LSPI learn quasi-optimal policies after only 5 to 10k dialogue turns, it means that only 1k to 2k dialogues were required to learn. The KTD algorithm learns dialogue policies before the Fitted-Q and LSPI don't even converge, this is why the curves for the these batch algorithms only start later.

Although the task addressed in this paper is relatively simple (3-slots), its size is similar in terms of state-action space to those presented in the state of the art dealing with RL for SDS optimization [Levin and Pieraccini, 1998; Levin *et al.*, 2000; Singh *et al.*, 1999; Lemon *et al.*, 2006]. Actually, scaling up in terms of dialogue task is not to be compared to scaling up in terms of RL problems. For example, the dialogue task described in [Gasic *et al.*, 2010; Jurcicek *et al.*, 2010] is a 12-slot dialogue problem (so 4 times more than the system described in this paper). Yet, the RL state space used by the authors to optimize the strategy is only composed of 2 continuous and 2 discrete dimensions which makes this problem of comparable size with the one addressed in this paper in terms of RL complexity. Moreover, the MDP framework is chosen and not the Partially Observable MDP (POMDP) framework like in [Gasic *et al.*, 2010; Jurcicek *et al.*, 2010]. Yet, the POMDP resumes to an MDP in these works through the summary space trick. So, the complexity of the example described in this paper is comparable to state-of-the-art but simpler to analyse, this is why we have opted for this task.

It is rare that in the SDS literature, algorithms are compared according to their sample-efficiency, that is the number of required dialogues to reach convergence toward a good dialogue policy. Learning curves are generally provided to show that some methods out-perform others but the actual number of dialogues required to learn is not the main focus. In [Levin and Pieraccini, 1998; Levin *et al.*, 2000] the authors mention that more than 700k dialogues were needed to learn the optimal policy on a 3 slots problem (for a database querying task). This seminal work encouraged researchers to build user simulation so as to generate enough data for RL algorithms to converge, this is probably why sample-efficiency has not been investigated so much. Yet, some recent works [Gasic *et al.*, 2010; Jurcicek *et al.*, 2010] have focused on sample-efficient RL algorithms. Nevertheless they use online and on-policy algorithms which also require user simulation, not only for expanding the datasets but simply to be run. Therefore, all the problems induced by simulation are still present [Schatzmann *et al.*, 2005].

In [Chandramohan *et al.*, 2010], the authors also propose the use of sample-efficient algorithms but batch ones while this paper proposes an online algorithm. The work reported in [Li *et al.*, 2009] is also about a sample-efficient batch algorithm. In that paper, the task is not more complex (although the vocabulary is, but the study reported here stands at the intention level) since a maximum of three slots are also requested by the system (most of dialogues only need one slot to be filled). The system is also helped by the introduction of human knowledge to pre-select actions. Also, [Li *et al.*, 2009] makes use of a user simulation method to generate different sets of data as was done in this paper which makes the work presented here suitable for comparison with the state of the art. Yet, simulation is not mandatory in the general case. In addition, the user simulation doesn't simplify the task to be learnt (which only depends on the state-action space size and the quality of the exploration done with the handcrafted

policy) and results are not losing generality.

Finally, the comparison of the reproducibility of the results when using Fitted-Q and LSPI shows that KTD is more stable. Indeed, the variance on the performances seems to be lower and decreasing with time with this new algorithm.

6 Conclusion

This contribution addressed the problem of rapid learning of optimal spoken dialogue management policies with reinforcement learning from online interaction while simply observing a poor policy behaving (off-policy learning). Several generalization schemes have been compared including batch and online learning algorithms. All the algorithms are off-policy meaning that they learn by observing a low-performance but acceptable policy. The experimental results show that the KTD-Q algorithm can learn a near optimal policy in few hundreds of dialog turns (corresponding to less than 100 dialogs). This means that an optimal policy could be directly learnt from collected data avoiding data expansion methods based on user modeling. Also better approximation schemes than linear parameterizations are envisioned such as neural networks. Moreover, similarly to GPTD [Gasic *et al.*, 2010], uncertainty maintained by the KTD framework can be used to actively explore the state-action space so as to fasten convergence. Finally, the Kalman filtering paradigm would allow introducing prior knowledge to handle partial observability by extending the parameter vector and adding evolution equations into the state-space model described in Section 3.

References

[Bellman, 1957] Richard Bellman. A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 1957.

[Chandramohan *et al.*, 2010] Senthilkumar Chandramohan, Matthieu Geist, and Olivier Pietquin. Optimizing Spoken Dialogue Management with Fitted Value Iteration. In *Interspeech'10*, Makuhari (Japan), 2010.

[Gasic *et al.*, 2010] Milica Gasic, Filip Jurcicek, Simon Keizer, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. Gaussian processes for fast policy optimisation of pomdp-based dialogue managers. In *SIGDIAL'10*, Tokyo, Japan, 2010.

[Geist and Pietquin, 2010a] Matthieu Geist and Olivier Pietquin. Kalman Temporal Differences. *Journal of Artificial Intelligence Research (JAIR)*, 39:489–532, October 2010.

[Geist and Pietquin, 2010b] Matthieu Geist and Olivier Pietquin. Managing Uncertainty within Value Function Approximation in Reinforcement Learning. In *Journal of Machine Learning Research, Workshop & Conference Proceedings (JMLR W& CP): Active Learning and Experimental Design*, Sardinia, Italy, 2010.

[Gordon, 1995] Geoffrey Gordon. Stable Function Approximation in Dynamic Programming. In *ICML'95*, 1995.

[Henderson *et al.*, 2008] James Henderson, Oliver Lemon, and Kallirroi Georgila. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics*, 2008.

[Jurcicek *et al.*, 2010] Filip Jurcicek, Blaise Thomson, Simon Keizer, Milica Gasic, François Mairesse, Kai Yu, and Steve Young. Natural Belief-Critic: a reinforcement algorithm for parameter estimation in statistical spoken dialogue systems. In *Interspeech'10*, Makuhari (Japan), 2010.

[Kalman, 1960] Rudolf Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[Lagoudakis and Parr, 2003] Michail Lagoudakis and Ron Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[Larsson and Traum, 2000] Staffan Larsson and David R. Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 2000.

[Lemon *et al.*, 2006] Oliver Lemon, Kalliroi Georgila, James Henderson, and Matthew Stuttle. An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system. In *EACL'06*, Morristown, NJ, USA, 2006.

[Levin and Pieraccini, 1998] Esther Levin and Roberto Pieraccini. Using markov decision process for learning dialogue strategies. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'98), Seattle, Washington*, 1998.

[Levin *et al.*, 2000] Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23, 2000.

[Li *et al.*, 2009] Lihong Li, Suhrid Balakrishnan, and Jason Williams. Reinforcement Learning for Dialog Management using Least-Squares Policy Iteration and Fast Feature Selection. In *InterSpeech'09*, Brighton (UK), 2009.

[Pietquin and Dutoit, 2006] O. Pietquin and T. Dutoit. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech and Language Processing*, 14(2):589–599, 2006.

[Pietquin *et al.*, 2011] O. Pietquin, M. Geist, S. Chandramohan, and H. Frezza-Buet. Sample-Efficient Batch Reinforcement Learning for Dialogue Management Optimization. *ACM Transactions on Speech and Language Processing*, 2011.

[Schatzmann *et al.*, 2005] Jost Schatzmann, Matthew N. Stuttle, Karl Weilhammer, and Steve Young. Effects of the user model on simulation-based learning of dialogue strategies. In *ASRU'05*, San Juan, Puerto Rico, 2005.

[Schatzmann *et al.*, 2006] Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The Knowledge Engineering Review*, 21(2):97–126, June 2006.

[Singh *et al.*, 1999] S. Singh, M. Kearns, D. Litman, and M. Walker. Reinforcement learning for spoken dialogue systems. In *NIPS'99*, Denver, USA, 1999.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 3rd edition, March 1998.

[Walker *et al.*, 1997] M. Walker, D. Litman, C. Kamm, and A. Abella. PARADISE: A framework for evaluating spoken dialogue agents. In *ACL'97*, Madrid (Spain), 1997.

[Williams and Young, 2007] Jason Williams and Steve Young. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):231–422, 2007.