OFF-POLICY LEARNING IN LARGE-SCALE POMDP-BASED DIALOGUE SYSTEMS

Lucie Daubigney[1,3], *Matthieu Geist*[1] *and Olivier Pietquin*[1,2] *

[1]IMS, Supélec (Metz, France)
[2] UMI 2958, GeogiaTech-CNRS (Metz, France)
[3]Project-Team MAIA, Loria (Nancy, France)

ABSTRACT

Reinforcement learning (RL) is now part of the state of the art in the domain of spoken dialogue systems (SDS) optimisation. Most performant RL methods, such as those based on Gaussian Processes, require to test small changes in the policy to assess them as improvements or degradations. This process is called *on policy* learning. Nevertheless, it can result in system behaviours 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*. Such methods usually fail to scale up and are thus not suited for real-world systems. In this contribution, a *sample-efficient*, *online* and *off-policy* RL algorithm is proposed to learn an optimal policy. This algorithm is combined to a *compact* non-linear value function representation (namely a multi-layers perceptron) enabling to handle large scale systems.

Index Terms— Spoken Dialogue Systems, Reinforcement Learning

1. INTRODUCTION

Spoken dialogue systems (SDS) are now commonly used for addressing various tasks like appointment scheduling, troubleshooting, tutoring, etc. When building an SDS, efficiency and naturalness are of great importance since SDS are interacting with humans who can be quickly annoyed when speaking to a machine. To obtain those characteristics, the dialogue manager (DM) of the SDS - responsible for taking decisions about what to say and when - should have an adapted behaviour which takes the user into account. More difficulty is brought by the speech recognizer and semantic analyser of the SDS which respectively transcripts and analyses what the user said. Indeed, the two components are error-prone which makes the actual user goal partially observable by the DM.

Handcrafting a strategy that will lead to the achievement of the task so as to satisfy the user, becomes rapidly untractable when the task is realistic. Indeed, it requires to identify all the possible situations which can be encountered during the course of a dialogue. So solutions have been proposed to automatically search for optimal strategies. We focus on those based on reinforcement learning (RL) [1]. RL is a machine learning technique that has been successfully applied to SDS optimisation [2, 3, 4, 5]. The idea of these algorithms is to learn an optimal strategy from interactions between the SDS and users, so as to optimise a numerical value (*reward*) related to user satisfaction. The optimal value function from which the optimal strategy can be derived is the one that associates the highest possible cumulative reward to each dialogue situation. Finding the optimal strategy thus resumes to the problem of learning the optimal value function. The major drawback of standard RL methods used so far in the field of SDS optimisation is to be very data demanding. The required number of actual interactions could not be collected in a tractable way. For more than one decade, this issue has been addressed by simulating dialogues so as to artificially generate enough data [6, 4, 7]. But this method introduces additional modeling bias which can lead to inadequacy of the learnt strategy with the behaviour of real users [8].

Efficient online algorithms have been proposed recently [9, 10]. These works report the use of *online* and *on-policy* algorithms requiring to permanently changing and testing the policy to learn. 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 behaviours of the dialogue manager. Moreover, these methods are based on linear approximations of the value function which makes their practical use in large-scale systems difficult. To scale-up, the dialogue context has to be represented in a compact way [11] which may lead to approximation errors.

In this paper, we propose to use the Kalman Temporal Differences (KTD) algorithm [12] to achieve *efficient online, off-policy* learning [13] in large-scale system. We also propose to use a *compact representation* to estimate the value function (based on Multilayer Perceptron) so as to minimize the impact of approximations. Off-policy learning will allow learning online from a behavioural strategy which is controlled After enough interactions, the DM can switch to the optimal strategy learnt to present it to the user.

2. DIALOGUE MANAGEMENT AS A CONTINUOUS MARKOV DECISION PROCESS

Dialogue management (DM) is actually a sequential decision making problem. From user acts (*observations*), the dialogue manager should choose and perform system acts (*actions*) in order to interact with the user in an efficient and natural way. User satisfaction is quantified by a *reward* provided at the end of a dialogue and computed as a linear combination of objective measures (such as the task completion, the dialogue duration, *etc.*).

Framed like this, DM can be cast as a Partially Observable Markov Decision Process (POMDP): decisions should be taken according to the full history of user and system acts. This history can be briefly and efficiently summarized into a single state by the hidden information state paradigm [5]. Thus, DM can be cast as a continuous state MDP [14].

Algorithms search for a policy π associating an action ($a \in A$) to each state ($s \in S$). The quality of the policy is quantified by a function called Q-function. This function associates to

*The authors want to thanks Steve Young, Miliça Gašić and the Cambridge Dialogue Systems Group for their help in using the CamInfo system. This work is partly funded by the INTERREG IVa project ALLEGRO and the Région Lorraine (France).

each state-action pair the expected cumulative reward after having started in this pair and then followed the policy π: $Q^{\pi}(s,a) = E[\sum_{i\geq 0}\gamma^i r_i|s_0 = s, a_0 = a, \pi]$. The factor γ is the discount factor, (s,a) the state-action pair and $(r_i)_{i\geq 0}$ the set of obtained rewards. The optimal Q-function, Q^*, is defined so that for all $(s,a) \in S \times A$, for all π, $Q^*(s,a) \geq Q^{\pi}(s,a)$. From Q^*, the optimal policy π^*, which is greedy relatively to it, can be derived: $\pi^*(s) = \arg\max_a Q^*(s,a)$. Notice that the Q-function allows comparing not only two policies but also two actions for a given state under a fixed policy. Usually, the state-action space is too large (often continuous) to allow an exact computation of the Q-function and an approximate representation $\hat{Q}(s,a)$ is mandatory. The representation is often parametric $\hat{Q}_{\theta}(s,a)$, with θ the set of parameters.

Two general schemes are used to compute the optimal Q-function. First, on-policy learning improves the policy being learnt incrementally, by iteratively computing the value of this policy then changing the policy by making it greedy according the computed Q-function. Because of the Markov property of the transition probabilities, the evaluation phase can be done thanks to the Bellman evaluation equation: $Q^{\pi}(s,a) = E_{s'|s,a}[R(s,a,s') + \gamma Q^{\pi}(s', \pi(s'))]$. The state s' is reached after having followed the action a returned by the policy π from s. This scheme is adopted by the GPTD algorithm used in [9]. The results provided by this algorithm will serve as a point of comparison for the experiments presented in this paper. Notice that the Bellman evaluation equation is linear which is mandatory for using Gaussian Processes (GP). GP constrain the parametrisation $\hat{Q}^{\pi}_{\theta}(s,a)$ to be linear as well.

The other scheme is off-policy learning. This method consists in directly computing the optimal value function $Q^*(s,a)$ using the non-linear Bellman optimality equation: $Q^*(s,a) = E_{s'|s,a}[R(s,a,s') + \gamma \max_{b\in A} Q^*(s',b)]$. The KTD framework [12] handling non-linearities, this equation can be solved directly and efficient off-policy learning is made possible [13]. In this paper, we propose to take advantage of this to use a non-linear parametrisation for $Q^*_{\theta}(s,a)$ to obtain a compact representation of the optimal value function and to scale-up to a richer state representation.

3. Q-FUNCTION APPROXIMATION

The Q-function (optimal or not) has to be approximated by $\hat{Q}_{\theta}(s,a)$ since the state space S is continuous and $Q : S \times A \to \mathbb{R}$. The representation for the Q-function can be linear or non-linear.

In the linear case, the Q-function is represented by $\hat{Q}_{\theta}(s,a) = \theta^T \Phi(s,a)$,where $\Phi(s,a)$ is a feature vector of size N (number of parameters) so that $\Phi = [\phi_1(s,a)...\phi_N(s,a)]^T$ is a predefined set of basis functions and θ are the associated weights.

If a non-linear representation is chosen, the Q-function can be represented by $\hat{Q}_{\theta}(s,a) = f_{\theta}(s,a)$, with f_{θ} being non-linear in the parameters θ. In this paper, we used an MLP which is known to be able to approximate any function if containing enough neurons on hidden layers. The vector θ thus contains the synaptic weights of the MLP.

We define $\mathcal{H}$ as the hypothesis space generated by the features $\Phi(s,a)$ or by $f_{\theta}(s,a)$. The $\mathcal{H}$ space has to be rich enough to contain the Q-function but the number of parameters should not be too high because of risks of a poor capacity of generalisation and computational costs.

4. EXPERIMENT

The results presented in the next section have been obtained with the CamInfo system [5], a large scale SDS developed to provide tourists informations about the Cambridge city (UK). The user request can contain up to 12 different attributes. All the results have been obtained with simulated users [15] because of the difficulty of having real data and because of the high variability between users possibly provided. This is not necessarily the case when only a sample of users is available. Speech understanding error are also simulated.

During the learning phase, a reward of $+20$ is given to the system at the end of the dialogue if the DM managed to satisfy the user request. A penalty of -1 is given at each system turn. The initial state space is built as described in [5]. A state is a vector containing two continuous variables (representing the 2-best confidence scores in the list of hypotheses of what the user said provided by the speech and semantic analysers) and two discrete ones (representing an hypothesis on the user goal and the user action associated with the top confidence score).

The results have been obtained by first letting the DM learn an estimation of the Q-function, $\hat{Q}_{learnt}(s,a)$. In the GPTD case, the Q-function estimation is improved at each step of the learning while in the KTD case, an estimation of the optimal Q-function is directly derived. Then the Q-function learnt will be used to test the greedy strategy $\pi_{learnt} = \arg\max_a \hat{Q}_{learnt}(s,a)$. With sufficient training data, the two algorithms are assumed to give the same optimal strategy: π_{learnt} should be equal to π^*. On the graphs presented, the average cumulated rewards got while using π_{learnt} is plotted for different sizes of training sets.

The learning phase supposes to use a behavioural strategy to explore the state space (issue known as the exploration/exploitation dilemma). The KTD and GPTD algorithms provide an estimation of the Q-function and some uncertainty information about the quality of the estimation ($\hat{\sigma}_Q$) [16, 9]. An approach where the agent makes a safe compromise between exploiting the already known information or exploring the state space has been studied in [17] on the same SDS and gave encouraging results. It will thus be used here. This approach is called *bonus-greedy* (inspired by [18]) and the choice of the next action is made according to: $a_{i+1} = \arg\max_a(\hat{Q}_i(s_{i+1},a) + \frac{\beta\hat{\sigma}_{Q_i}(s_{i+1},a)}{\beta_0+\hat{\sigma}_{Q_i}(s_{i+1},a)})$.

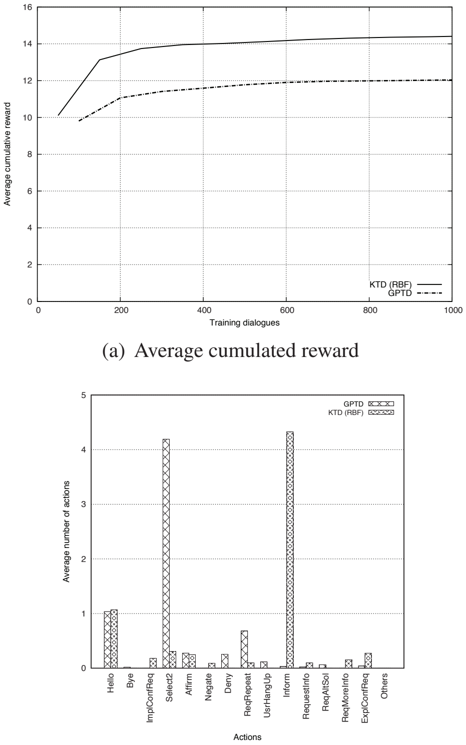
5. RESULTS

The GPTD algorithm using a linear parametrisation based on a dictionary built during the learning phase [19] is compared to KTD using at first a linear parametrisation and then a non-linear one.

First, the comparison is made in an almost noisyless environment (the recognition error rate is set to 10%). The number of parameters is about 300 for the GPTD algorithm and 144 for the KTD one. The vector of parameters for the latest, defined for all $(s,a) \in S \times A$ is: $\Phi^T(s,a) = [\delta_{a,a_1}\phi^T(s,a_1), ..., \delta_{a,a_{12}}\phi^T(s,a_{12})]$, with $\phi^T(s,a) = [1, \varphi^1_1, \varphi^1_2, \varphi^1_3, \varphi^2_1, \varphi^2_2, \varphi^2_3, I_1, I_2](s,a)$ and $\delta_{a,a'} = 1$ if $a = a'$ else $\delta_{a,a'} = 0$. Three Gaussians φ are used per continuous dimension and two integers, I_1, I_2 are used for discrete ones. The DM can choose actions among 12 meta-actions $(a_1, a_2, ..., a_{12})$ which can result into 22 different actual actions for the user (see [5]).

The results on Fig. 1(a) show that KTD outperforms GPTD. KTD policy results in nearly 2.5 turns less per dialogue. In both cases, the solution found is imperfect because of the compression of the state space as explained in [5]. The speech and semantic analyzers return a list of hypotheses with a confidence score associated. Some information is extracted from the pool of hypotheses to be given to the DM since all the hypotheses cannot be taken into account. But in the GPTD setting described in [9], information is also lost because some clustering of the state space is performed. It is not the case with KTD where all the states encountered are taken into account for the estimation of the Q-function.

On Fig. 1(b), the average frequency of each of the actions performed by the dialogue manager during the testing phase is shown. The "Others" action regroups 7 others actions which are not used here. By studying the actions proposed, the differences seen in the graphes can be explained and the two algorithms compared. The two policies found do not propose the same main action: GPTD asks for the user to select between two propositions ("Select2" action) while it provides the user with direct information in KTD ("Inform" action). GPTD asks for the user to repeat quite oftenly ("ReqRepeat" action) while KTD prefers to ask for explicit confirmation ("ExplConfReq" action). It is easier to recognize a yes/no answer than a complete sentence. KTD asks sometimes for new information while implicity confirming another ("ImplConfReq" action) which can shorten the length of the dialogue. GPTD proposes sometimes partially right solution ("Deny" action) which is linked to the use of the "Select2" action.



(a) Average cumulated reward

(b) Frequency of actions

Fig. 1. GPTD and KTD with linear parametrisation comparison.

A non-linear parametrisation based on a Multi Layers Perceptron (MLP) is now introduced. One hidden layer of 8 neurons is used ($N_{H_1} = 8$). The number of input neurons of the MLP is determined by the fact that to parts of each pair $(s, a) \in (S \times A)$ must be associated a unique binary combination, given that no metric is defined over the action space. The number of inputs, N_I is: for the state space, 2 neurons for the continuous components (2 top scores), 6 for the recognized goals, 22 for the user actions; for the action space, 12 neurons are needed. So $N_I = 2{+}6{+}22{+}12 = 42$. The output layer of the MLP contains one neuron (the value of the Q-function). The number of parameters is: $N_I{\cdot}N_{H_1}{+}(N_{H_1}{+}1){\cdot}1 = 42{\cdot}8{+}9 = 345$.

The parametrisation based on an MLP is interesting since even if the input space is bigger, the number of parameters needed to correctly represent the function to estimate does not become huge. For example, if a discrete value taking N different values is added to the state space, the number of parameters is multiplied by N while with the neural network approach here, only $N_{H_1} \cdot N$ parameters are added to the previous parametrisation. In the case of a continuous variable, only one input neuron is added. This property can be used to increase the state space size and enrich the state representation. It will thus avoid losing some easily available information.

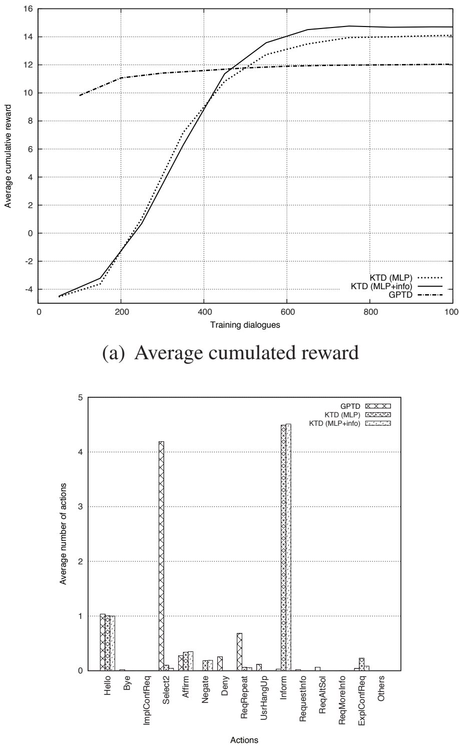
A continuous dimension is thus added by taking into account the third confidence score as well as a discrete dimension which is the user action associated with the second top confidence score. The new state space has now six components. The number of inputs is $N_I = 42 + 22 + 1 = 65$ so the number of parameters is: $65 \cdot 8 + 9 = 529$. To compare, if a RBF approach was used, the number of parameters would be about $(1 + 3^3) \cdot 6 \cdot 22 \cdot 22 \cdot 12 = 975.744$, with only 3 Gaussians in each of the continuous dimensions.

The performance of the KTD approach in an environment with little noise (error recognitition rate set to 10%), with this non-linear parametrisation and either the initial state space or the new state space has been compared to the GPTD one in Fig. 2(a). The GPTD approach is used with the initial state space because of its unability to scale up. The results obtained with the KTD algorithm are better than the one got with GPTD considering that sufficient training data are provided.When an MLP is used, the curve steps because of the intrinsic property of the MLP: during the learning, the Q-function is either not approximated (when not enough training data are provided) or correctly approximated (after a sufficient number of data are provided). The difference when information is added to the state space is not obvious. Adding information when there is little noise is not very interesting since the speech and semantic analysers are confident with what they recognize. So the list of hypotheses they make is not very long and all the information is contained in the very top ones. In Fig. 2(b) are compared the frequencies of the actions returned by the DM while testing the three policies. The histograms are quite similar to those obtained in Fig. 1(b).

Adding information should be more interesting in noisy situations since the list of hypotheses should be longer and the confidence score associated should be more uniformly distributed among hypotheses. For that reason, a learning has been made in a very noisy situation (recognition rate sets to 50%). The average number of parameters in GPTD approach is about 700. This number which is the size of a dictionary built online, is bigger in noisy environments. This dictionary is larger in that case since a wider part of the state space is visited. The results are presented Fig. 3. While there was no remarkable difference between the runs when little noise was set, now a difference exists. In Fig. 3(a), the MLP approach with initial state space is still better than the GPTD approach. Now in the case where additional information is brought, the KTD approach with enriched state space gives better results: the average length of a dialogue is shorten by 2 turns. In Fig. 3(b), the frequencies of the actions proposed are presented. The trend between the GPTD and the KTD algorithms are the same than in the little noisy environment. But explicit requests are more frequent when less information is present.

6. CONCLUSION

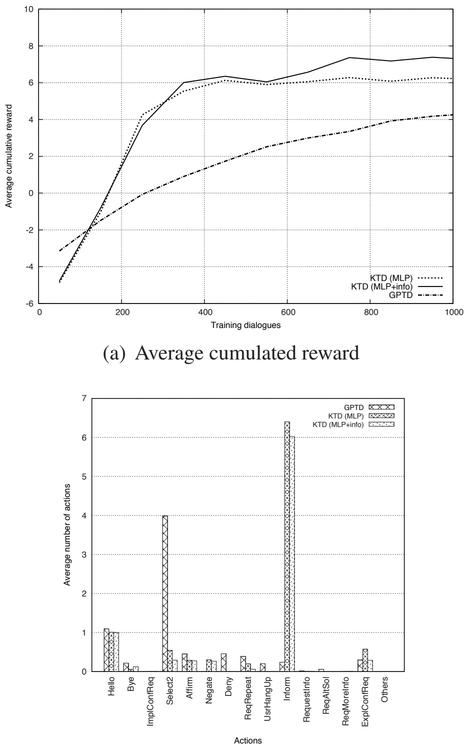
In this paper, we proposed the use of a Kalman Temporal Differences based algorithm [12] to learn efficiently in an off-policy manner a strategy for a large scale dialogue system. A linear parametrisation has fisrt been used to represent the Q-function. Then, because the KTD framework is able to handle non-linear parametrisations,



(a) Average cumulated reward

(b) Frequency of actions

Fig. 2. GPTD and KTD (with non-linear parametrisation) comparison (10% recognition errors).



(a) Average cumulated reward

(b) Frequency of actions

Fig. 3. GPTD and KTD (with non-linear parametrisation) comparison (50% recognition errors).

it has been associated to a compact value function approximator in the form of a Multi Layer Perceptron. Thanks to this compact representation, the dialogue state representation could be enriched to include additional information about the distribution of confidence scores over hypothesis. It is interesting since by taking more information into account, less modeling bias is introduced. This enriched state representation showed to outperform current algorithms like the Gaussian Process Temporal Differences algorithm and more robust dialogue strategies could be computed.

In the future, richer state representations will be tested since the increase in the number of parameters is not prohibitive in the case of a MLP-based value function approximation.

7. REFERENCES

[1] R.S. Sutton and A.G. Barto, *Reinforcement learning: An introduction*, The MIT press, 1998.

[2] S. Singh, M. Kearns, D. Litman, and M. Walker, “Reinforcement learning for spoken dialogue systems,” in *Proc. NIPS’99*, 1999.

[3] E. Levin, R. Pieraccini, and W. Eckert, “A stochastic model of human-machine interaction for learning dialog strategies,” *IEEE TSAP*, 2000.

[4] O. Pietquin and T. Dutoit, “A probabilistic framework for dialog simulation and optimal strategy learning,” *IEEE TSAP*, 2006.

[5] S. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, “The hidden information state model: A practical framework for POMDP-based spoken dialogue management,” *Computer Speech & Language*, 2010.

[6] W. Eckert, E. Levin, and R. Pieraccini, “User modeling for spoken dialogue system evaluation,” in *Proc. ASRU’97*, December 1997.

[7] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young, “A survey of statistical user simulation techniques for rl of dialogue management strategies,” *The Knowledge Engineering Review*, 2006.

[8] J. Schatzmann, M. N. Stuttle, K. Weilhammer, and S. Young, “Effects of the user model on simulation-based learning of dialogue strategies,” in *Proc. of ASRU’05*, 2005.

[9] M. Gašić, F. Jurčíček, S. Keizer, F. Mairesse, B. Thomson, K. Yu, and S. Young, “Gaussian processes for fast policy optimisation of POMDP-based dialogue managers,” in *Proc. of SIGDIAL 11*, 2010.

[10] F. Jurcicek, B. Thomson, S. Keizer, M. Gasic, F. Mairesse, K. Yu, and S. Young, “Natural Belief-Critic: a reinforcement algorithm for parameter estimation in statistical spoken dialogue systems,” in *Interspeech’10*, 2010.

[11] J. Williams and S. Young, “Scaling up POMDPs for dialogue management: the summary POMDP method,” in *Proc. of ASRU*, 2005.

[12] M. Geist and O. Pietquin, “Kalman Temporal Differences,” *JAIR*, 2010.

[13] O. Pietquin, M. Geist, and S. Chandramohan, “Sample Efficient On-line Learning of Optimal Dialogue Policies with Kalman Temporal Differences,” in *Proc. of IJCAI 2011*, 2011.

[14] J. Williams and S. Young, “Partially observable Markov decision processes for spoken dialog systems,” *Comp. Speech and Language*, 2007.

[15] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young, “Agenda-based user simulation for bootstrapping a pomdp dialogue system.,” in *HLT/NAACL 2007*, 2007.

[16] M. Geist and O. Pietquin, “Managing Uncertainty within the KTD Framework,” in *Proc. of the AL&E workshop*, 2011, JMLR C&WP.

[17] L. Daubigney, M. Gasic, S. Chandramohan, M. Geist, O. Pietquin, and S. Young, “Uncertainty management for on-line optimisation of a POMDP-based large-scale spoken dialogue system,” in *Proc. of Interspeech 2011*, 2011.

[18] J. Z. Kolter and A. Y. Ng, “Near-Bayesian Exploration in Polynomial Time,” in *Proc. of ICML 09*, 2009.

[19] Y. Engel, S. Mannor, and R. Meir, “Reinforcement Learning with Gaussian Processes,” in *Proc of ICML 05*, 2005.