Uncertainty management for on-line optimisation of a POMDP-based large-scale spoken dialogue system

Lucie Daubigney[1], *Milica Gašić*[2], *Senthilkumar Chandramohan*[1], *Matthieu Geist*[1], *Olivier Pietquin*[1,3], *Steve Young*[2]

[1]Supélec - Metz Campus, IMS Research Group, France
[2] Cambridge University, Engineering Department, UK
[3]UMI 2958 (CNRS - GeorgiaTech), France

[1,3]firstname.lastname@supelec.fr
[2]{mg436,sjy}@eng.cam.ac.uk

Abstract

The optimization of dialogue policies using reinforcement learning (RL) is now an accepted part of the state of the art in spoken dialogue systems (SDS). Yet, it is still the case that the commonly used training algorithms for SDS require a large number of dialogues and hence most systems still rely on artificial data generated by a user simulator. Optimization is therefore performed off-line before releasing the system to real users. Gaussian Processes (GP) for RL have recently been applied to dialogue systems. One advantage of GP is that they compute an explicit measure of uncertainty in the value function estimates computed during learning. In this paper, a class of novel learning strategies is described which use uncertainty to control exploration on-line. Comparisons between several exploration schemes show that significant improvements to learning speed can be obtained and that rapid and safe online optimisation is possible, even on a complex task.

Index Terms: spoken dialogue systems, reinforcement learning

1. Introduction

Spoken Dialogue Systems (SDS), and especially task-oriented SDSs, are now very common and are often encountered in everyday life (hotline, call routing *etc.*). A key requirement when building such a dialogue system is to design a dialogue manager that fulfils the user requests in a way which is robust, efficient and natural. To achieve optimality, dialogue strategies should handle speech and language processing errors robustly, exploiting implicit and explicit confirmation as well as disambiguation sub-dialogues as appropriate. Dialogue management can therefore be seen as a sequential decision making problem where decisions about which dialogue act should be output in a given context (dialogue state) should be taken so as to maximize some optimality criterion. Sequential decision making is often addressed by Reinforcement Learning (RL) [1] which is a statistical machine learning method that has been successfully applied to SDS during the last decade [2, 3, 4, 5]. Optimizing dialogue management by means of RL requires casting the problem into a Markov Decision Process (MDP) [3] or a Partially Observable MDP (POMDP) [5]. In the latter case, decisions are not taken according to a single inferred context but according to a distribution over all possible contexts. This makes the learning process more robust to speech recognition and understanding errors but at the cost of significantly more complex dialogue policies.

Although the (PO)MDP framework is appealing for SDS optimisation, standard RL algorithms need large amounts of data to converge to an optimal strategy. Dataset expansion by means of user simulation has thus been intensively studied during the last 15 years [6, 4, 7]. Yet, user simulation induces a modeling bias which is hard to predict [8] and can lead to sub-optimal strategies after learning. In the meantime, RL research has made progress. In particular, batch algorithms that can learn from small amounts of fixed data have recently been shown to be applicable to SDS [9]. Furthermore, the examples used for learning, whether simulated or real, may not represent all types of users. The ability of a system to adapt during actual use would greatly mitigate this. Recently, online learning with an efficient RL algorithm, namely GP-SARSA [10], has been proposed to optimize dialogue management strategies [11]. GP-SARSA makes use of Gaussian processes [12] to approximate the state-action value function from which an optimal policy can be easily derived.

Several approaches to on-line policy optimisation can be envisioned depending on the way the policy is modified during learning. If the initial policy is modified after each interaction with a user, the approach is called *on-policy*. Whereas when a policy is improved by using interactions collected with another policy, the approach is called *off-policy*. While off-policy learning offers the possibility to use a non-optimal but acceptable policy during learning, it is not obvious how to identify the best moment to disable exploration and switch to the learnt policy. On the other hand, on-policy learning (as in [11]) allows the policy to be updated immediately after each interaction but requires the user to suffer changes made to the policy during exploration, even if the changes are undesirable. Ideally, therefore, learning should be as quick as possible to minimise unacceptable changes to the policy affecting the users. However, this then raises the well-known exploration vs exploitation dilemma: should the system exploit its current knowledge and stop learning with the risk of being suboptimal or should it continue exploring different possible strategies to learn a better one with the risk of acting dangerously.

In this paper, we propose to use the uncertainty information relating to intermediate value estimates (namely the Q-function) computed by the Gaussian Process for RL method to build a sample-efficient online learning algorithm. A key ad-

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).

vantage of the method is that exploring actions are selected not only because they are uncertain but also because they are expected not to be harmful. The results are compared with [11].

The rest of the paper is organized as follows. In Section 2, the problem of casting SDS optimization into the POMDP framework is presented. The following section then proposes several exploration schemes based on the uncertainty of the value estimates. In Section 4 some results are presented using an operational dialogue system for tourist information. Finally, conclusions are drawn and perspectives offered.

2. Dialogue Management as a POMDP

Dialogue management (DM) can be seen as a sequential decision making problem. The decision maker is the dialogue manager. From user acts (*observations*), it should choose and perform a system act (*actions*) in order to satisfy the user in an efficient and natural way. This satisfaction is quantified by a *reward* provided at the end of a dialogue and computed as a mixture 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) where each possible history is represented by a state and the decision is taken according to the distribution over all states. In practice, the exact implementation of a POMDP is intractable but there are efficient methods for approximating the state. In particular, in the hidden information state paradigm [5], the state distribution is reduced to a summary space which has the form of a continuous state MDP. In what follows, the term state refers to a state of this MDP.

DM decisions are defined by a policy π associating an action to each state. Its quality is quantified by a so-called Q-function that gives the expected cumulative reward for starting with a given state-action pair and then following the policy π:

$$Q^{\pi}(s,a) = E\Big[\sum_{i\geq 0}\gamma^i r_i \Big| s_0 = s, a_0 = a, \pi\Big],$$

γ being a discount factor, (s,a) the state-action pair and $(r_i)_{i\geq 0}$ the set of obtained rewards. The optimal policy (π^*) is greedy wrt to its Q-function ($Q^*(s,a)$): $\pi^*(s) = \arg\max_a Q^*(s,a)$. The problem of finding an optimal policy therefore reduces to learning the optimal Q-function. The Q-function not only allows two policies to be compared, but it also allows the possible actions in any given state to be compared under a fixed policy. Usually, the state-action space is too large to allow an exact computation of the Q-function and approximation is mandatory.

Here we are interested in learning an optimal control policy while interacting with the user i.e. we require on-line and on-policy learning. There are usually two steps: estimating the Q-function of the followed policy (with SARSA [1] for example) and choosing actions according to this estimated Q-function (the control part).

Given an appropriate Q-function approximation algorithm, the problem of choosing actions according to the current Q-function estimate requires a choice between exploration and exploitation. At each interaction, the RL agent should choose between acting according to its current (imperfect) representation of the world (here the estimated Q-function) and performing some exploration action which although suboptimal according to the current estimate might in fact lead to a higher reward. A classical but crude scheme is the use of an ϵ-greedy policy whereby a greedy action is taken wrt to the estimated Q-function ($\arg\max_a Q(s,a)$) with probability (w.p.) $1-\epsilon$, and a random one w.p. ϵ. If learning from interactions with real users is envisioned, the choice between exploration and exploitation is crucial. First, learning should be fast: the DM should improve quickly and be able to adapt rapidly to users. Second, learning should be safe: the DM should not choose repetitively bad actions. This suggests that choosing actions purely randomly as done by the ϵ-greedy policy may not be wise.

3. Gaussian Processes and Exploration

The underlying idea of this paper is that if one has access to some uncertainty information about the estimated Q-function, then it should provide useful information for handling the choice between exploration and exploitation. For example, assume that two actions are possible for a given state, one (say a_1) having a higher estimated Q-value than the other (a_2). A greedy agent would choose a_1. But assume also that confidence intervals for these estimates are available. If the uncertainty about a_2 is high, this action may in fact be better than a_1, and so it may be worth trying. On the other hand, if the confidence interval is small, one should choose a_1. This is an instantiation of active learning where the utility function uses uncertainty information [13]. In this paper, we propose exploration schemes which make this choice automatically.

In order to provide the required uncertainty information, we use an extension of GP-based temporal difference learning called GP-SARSA [10]. The use of GP-SARSA for fast learning in spoken dialogue systems was first described in [11] and the same system is used here. The underlying principle of GP-SARSA is to model the state-action value function as a Gaussian Process (that is a set of jointly Gaussian random variables, these random variables being the values of each state-action pair). A generative model linking rewards to values via the sampled Bellman evaluation operator and an additive noise is set: $r_i = \hat{Q}(s_i,a_i) - \gamma\hat{Q}(s_{i+1},a_{i+1}) + n_i$, n_i being the additive noise. The Gaussian distribution of a state-action pair value conditioned on past observed rewards is computed by performing Bayesian inference and the value of this state-action pair is estimated as the mean of this Gaussian distribution. The associated variance quantifies the uncertainty. As in any Bayesian method, a prior must be defined. For GP-SARSA, this is done through the choice of a kernel function which defines the prior correlations of Q-function values.

The GP-SARSA algorithm provides at each step i an estimate $\hat{Q}_i(s,a)$ of the Q-function as well as an associated standard deviation $\hat{\sigma}_i(s,a)$. The ϵ-greedy policy does not use this uncertainty information: the greedy action $a_{i+1} = \arg\max_a \hat{Q}_i(s_{i+1},a)$ is chosen w.p. $1-\epsilon$ and a (uniform) random action otherwise. The obtained transition is then used to learn the new estimate $\hat{Q}_{i+1}$ (and $\hat{\sigma}_{i+1}$). Under some conditions, notably a decaying exploration factor, it converges to the optimal policy. However, it can be rather slow and unsafe during the exploration stage.

In [11] an *informed exploration* policy is introduced and this is used here as a baseline. It chooses the greedy action (wrt to the estimated Q-function) $a_{i+1} = \arg\max_a \hat{Q}_i(s_{i+1},a)$ w.p. $1-\epsilon$ and another greedy action (wrt to the computed variance) $a_{i+1} = \arg\max_a \hat{\sigma}_i^2(s_{i+1},a)$ w.p. ϵ:

$$a_{i+1} = \begin{cases} \arg\max_a \hat{Q}_i(s_{i+1},a) \text{ w.p. } 1-\epsilon \\ \arg\max_a \hat{\sigma}_i^2(s_{i+1},a) \text{ w.p. } \epsilon \end{cases} \quad (1)$$

Therefore, the underlying idea is to act greedily wrt the estimated state-action value function, and to occasionally choose

an exploratory action. By choosing the less certain action, this scheme actually chooses the action which provides more information. Moreover, the exploration parameter ϵ is decayed to zero as learning progresses, so this scheme tends to the greedy policy. This improves over the classical ϵ-greedy policy but with the possible drawback that the exploration stage does not take the estimated value into account. Therefore, an action with a very low estimated value (which is possibly a "dangerous" action) can be chosen repeatedly, as long as it has a higher uncertainty than other actions (even if the difference is slight). To avoid this drawback, we propose to use of a different form of exploration scheme.

The *confident-greedy* policy consists of acting greedily according to the upper bound of an estimated confidence interval [14]. For a tabular representation (for which the confidence interval width is proportional to $\frac{1}{\sqrt{n(s,a)}}$, where $n(s,a)$ is the number of visits to the considered state-action pair), some PAC (Probably Approximately Correct) guarantees can be provided [15]. In the case of continuous state spaces (which occur in SDS optimization), the state-action pairs are uncountable, this approach does not hold. Here, standard deviation is provided by a Bayesian method, the prior is given and the posterior is computed. The distribution being Gaussian by assumption, the confidence interval width is proportional to the estimated standard deviation (which is actually true for any distribution, according to the Bienaymé-Tchebychev concentration inequality). Let α be a free positive parameter. We define the confident-greedy policy as:

$$a_{i+1} = \arg\max_a (\hat{Q}_i(s_{i+1}, a) + \alpha \hat{\sigma}_i(s_{i+1}, a)) \qquad (2)$$

This strategy favors less certain actions if they correspond to the upper bound of the confidence interval. On the other hand, if for a given state all standard deviations are equal, than the agent will act greedily wrt $\hat{Q}_i$. Notice also that the variance provided by GP-SARSA decreases as the prediction accuracy of rewards improves and as actions are more and more explored. A similar strategy has been shown to be efficient in model-based reinforcement learning using Gaussian Processes [13].

The second approach we consider is the *bonus greedy* policy, inspired by [16]. Using a Bayesian argument, it acts greedily wrt the estimated Q-function plus a bonus, this bonus being proportional to the inverse of the number of visits to the state-action pair of interest. As $\frac{1}{\sqrt{n(s,a)}}$ is proportional to the standard deviation in a frequentist approach, we interpret $\frac{1}{n(s,a)}$ as a variance. The proposed bonus-greedy policy therefore uses a variance-based bonus and is defined as:

$$a_{i+1} = \arg\max_a (\hat{Q}_i(s_{i+1}, a) + \beta \frac{\hat{\sigma}_i^2(s_{i+1}, a)}{\beta_0 + \hat{\sigma}_i^2(s_{i+1}, a)}) \qquad (3)$$

where β_0 and β are free parameters. This strategy also favors less certain actions and tends to be greedy wrt the estimated state-action value function if variances are close to each other.

4. Experiments

Experiments have been conducted using the Hidden Information State (HIS) dialogue manager [5]. The task concerns a tourist information system which assists users to find a venue in Cambridge using up to 12 attributes. The algorithm used for the dialogue management optimisation is GP-SARSA with a polynomial kernel for the summary state space and a Dirac kernel for the action space; it uses an identical set-up for the Q-function approximator as in [11]. To obtain sufficient dialogues, the dialogue manager interacts at the intention level with a simulated user [17]. The simulated user consists of a goal and an agenda. The goal ensures that the user simulator exhibits consistent, goal-directed behaviour. The role of the agenda is to elicit the dialogue acts needed to complete the goal. The speech understanding error rate is set to 10% by using an error simulator, which randomly confuses the semantic concepts that occur in user dialogue acts. A positive reward of $+20$ is given at the end of the dialogue if the DM managed to fulfil the user request and a penalty of -1 is applied per system turn to encourage short dialogues. The efficiency of the different exploration/exploitation schemes was compared with the *informed exploration* scheme described in Section 3.

The different schemes are first compared using the same approach as in [11]. In each case, a policy is trained using a specific number of dialogues and the policy is tested (without exploration). The parameters used in the two new exploration schemes were set as follows: $\alpha = 12$, $\beta = 1000$ and $\beta_0 = 100$. These were chosen empirically to give good results but they are quite easy to tune since their order of magnitude can be deduced from the variance approximation of the Q function. The parameter $\epsilon_0 = 0.1$ was set at the start of training in the informed exploration scheme and then decreased during the learning phase according to the number of dialogues experienced.

Figure 1 shows the reward achieved as a function of the number of training dialogues. As can be seen, the confident greedy and bonus greedy algorithms both learn faster initially but eventually converge to similar values. However, the confident greedy scheme is consistently better than the bonus greedy scheme.

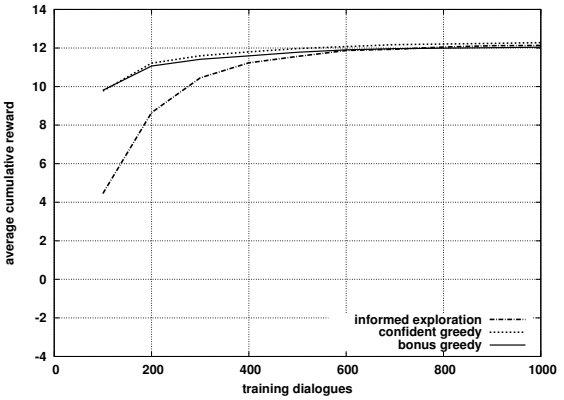

Figure 1: *Performance of the trained policies (greedy setting). To compute the average reward, each* 100 *dialogues,* 100 *policies were trained and tested in a greedy way* 1000 *times each.*

The above results allow the performance of the trained policies to be assessed but they do not take account of the fact that during learning, exploration may result in some dialogues having very poor rewards. Whilst this is acceptable for off-line training, it may not be acceptable to users during on-line training. To assess the detrimental impact of poor rewards during on-line training, Figure 2 shows the rewards actually obtained during training as a function of the number of dialogues processed. As can be seen, the two newly proposed algorithms yield much better rewards during training. This is because they avoid subjecting the user to bad exploratory actions. A further advantage of the new on-line learning algorithms is that they

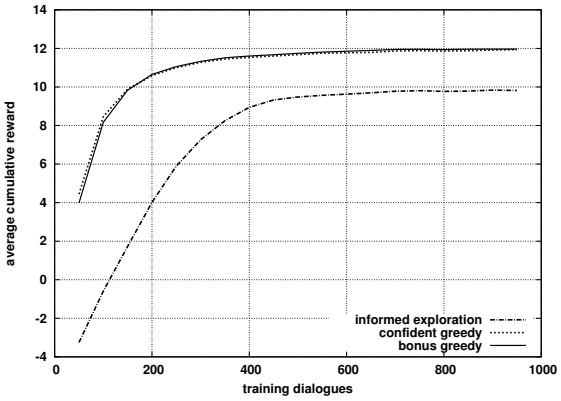

Figure 2: *Performances of the policies (online setting). The curves are obtained for an average over* 1000 *training trials. An average is made with a sliding window of* 100 *points width each* 50 *points.*

avoid the need to decide when a trained policy is sufficiently good that exploration can be disabled. Indeed, they allow exploration to be resumed as needed if and when user behaviour changes as indicated by increasing confidence intervals. Thus, these algorithms allow for continuous adaptation during operation.

The above results show that improvements are obtained when the estimated Q values are augmented by uncertainty information when selecting actions. First, when learning on-line, the asymptotic performance of the new schemes is higher (around 12 v.s 10) because they do not suffer from the poor rewards resulting from random exploration.

Second, the number of samples needed to reach the same quality of policy is smaller (about 150 dialogues compared to 600). This is an advantage when interacting with real users, since the quicker a good policy is learnt the less the user will be annoyed by poor behaviour of the dialogue manager. Furthermore, the ability to learn policies quickly will obviate the need for training using a user simulator. This is a significant benefit, since good user simulators are expensive to build.

5. Conclusion and Perspectives

In this paper, we have proposed two novel exploration schemes to learn *on-line* and *on-policy* an optimal dialogue management strategy using reinforcement learning. The key feature of the new schemes is that they augment the Q function value of each state-action pair with an estimate of its uncertainty. This has the effect that the dialogue manager will choose the action which leads to the highest reward when it is confident about its estimate of that reward, otherwise it will choose the action about which it is least certain. This does not mean that the dialogue manager will never choose bad actions, but it does reduce the frequency with which it does so. The net result is faster learning and a higher reward during learning. This makes the new schemes more suitable for continuous on-line adaptation compared to conventional ϵ-greedy schemes.

The new exploration schemes were evaluated in this paper using GP-SARSA. In the future, alternative sample-efficient reinforcement learning algorithms such as in [18, 19, 20] will be tested with the aim of improving the learning speed further.

6. References

[1] R. Sutton and A. 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 Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 11–23, 2000.

[4] O. Pietquin and T. Dutoit, "A probabilistic framework for dialog simulation and optimal strategy learning," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 2, pp. 589–599, March 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*, vol. 24, no. 2, pp. 150–174, 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 reinforcement-learning of dialogue management strategies," *The Knowledge Engineering Review*, vol. 21, no. 2, pp. 97–126, June 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 *Proceedings of ASRU'05*, December 2005.

[9] 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, accepted for publication - 24 pages.

[10] Y. Engel, S. Mannor, and R. Meir, "Reinforcement Learning with Gaussian Processes," in *Proceedings of the International Conference on Machine Learning (ICML 05)*, 2005.

[11] 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 *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, 2010, pp. 201–204.

[12] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, January 2006.

[13] M. Deisenroth, C. Rasmussen, and J. Peters, "Gaussian Process Dynamic Programming," *Neurocomput.*, vol. 72, no. 7-9, pp. 1508–1524, 2009.

[14] L. P. Kaelbling, *Learning in embedded systems*. MIT Press, 1993.

[15] A. L. Strehl and M. L. Littman, "An Analysis of Model-Based Interval Estimation for Markov Decision Processes," *Journal of Computer and System Sciences*, 2006.

[16] J. Z. Kolter and A. Y. Ng, "Near-Bayesian Exploration in Polynomial Time," in *international conference on Machine learning (ICML 09)*. New York, NY, USA: ACM, 2009.

[17] 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*. Association for Computational Linguistics, 2007, pp. 149–152.

[18] M. Geist and O. Pietquin, "Kalman Temporal Differences," *Journal of Artificial Intelligence Research (JAIR)*, vol. 39, pp. 483–532, 2010.

[19] ——, "Statistically Linearized Least-Squares Temporal Differences," in *Proceedings of the IEEE International Conference on Ultra Modern Control systems (ICUMT 2010)*. Moscow (Russia): IEEE, October 2010, 8 pages.

[20] O. Pietquin, M. Geist, and S. Chandramohan, "Sample efficient on-line learning of optimal dialogue policies with kalman temporal differences," in *International Joint Conference on Artificial Intelligence (IJCAI 2011)*, Barcelona, Spain, July 2011.