CLASSiC

D1.5: Online adaptation of dialogue systems

F. Jurčíček, M. Gašić, S. Young, R. Laroche, G. Putois,
M. Geist and O. Pietquin

Distribution: Public

CLASSiC
Computational Learning in Adaptive Systems for Spoken Conversation
216594 Deliverable 1.5

February 2011

Project funded by the European Community
under the Seventh Framework Programme for
Research and Technological Development

The deliverable identification sheet is to be found on the reverse of this page.

Project ref. no.	216594
Project acronym	CLASSiC
Project full title	Computational Learning in Adaptive Systems for Spoken Conversation
Instrument	STREP
Thematic Priority	Cognitive Systems, Interaction, and Robotics
Start date / duration	01 March 2008 / 36 Months

Security	Public
Contractual date of delivery	M36 = February 2011
Actual date of delivery	February 2011
Deliverable number	1.5
Deliverable title	D1.5: Online adaptation of dialogue systems
Type	Report
Status & version	Draft 0.92
Number of pages	10 (excluding front matter)
Contributing WP	1
WP/Task responsible	WP1, leader UCAM
Other contributors	
Author(s)	UCAM: Filip Jurčíček, Milica Gašić, Steve Young - FT: Romain Laroche, Ghislain Putois - SUPELEC: Matthieu Geist and Olivier Pietquin
EC Project Officer	Philippe Gelin
Keywords	online adaptation, Gaussian processes, actor critic reinforcement learning, spoken dialogue management, partially observable Markov decision processes

The partners in CLASSiC are:

Heriot-Watt University	HWU
University of Cambridge	UCAM
University of Geneva	GENE
Ecole Superieure d'Electricite	SUPELEC
France Telecom/ Orange Labs	FT
University of Edinburgh HCRC	EDIN

For copies of reports, updates on project activities and other CLASSiC-related information, contact:

The CLASSiC Project Co-ordinator:
Dr. Oliver Lemon
School of Mathematical and Computer Sciences (MACS)
Heriot-Watt University
Edinburgh
EH14 4AS
United Kingdom
O.Lemon@hw.ac.uk
Phone +44 (131) 451 3782 - Fax +44 (0)131 451 3327

Copies of reports and other material can also be accessed via the project's administration homepage, `http://www.classic-project.org`

Contents

Executive summary

This document is a report on online adaptation of dialogue systems (deliverable 1.5), due at month 36 of the CLASSiC project.

It consists of four contributions. First, it demonstrates fast policy adaptation using the GP-SARSA algorithm applied to Hidden Information State (HIS) dialogue manager. Second, it describes online adaptation of dialogue model parameters using the NBC algorithm within the Belief Update of Dialogue State (BUDS) dialogue manager. Third, it proposes the Kalman Temporal Differences algorithm for management of uncertainty in estimate of the optimal value function. Finally, it details optimisation techniques for industrial spoken dialogue systems based on compliance-based reinforcement learning.

Work related to this deliverable has been published in Gašić et al. (2010), Jurčíček et al. (2010b), Laroche et al. (2010b), and Geist and Pietquin (2010, 2011).

1 Introduction

Spoken dialogue systems allow a human user to interact with a machine using voice as the primary communication medium. A typical dialogue system consists of a speech understanding component, a dialogue manager, and a speech generation component. Speech understanding usually consists of a speech recogniser and a semantic parser, and speech generation requires a natural language generator and a speech synthesiser.

In recent years, Partially Observable Markov Decision Processes (POMDPs) have been proposed as a principled way of modelling the uncertainty in spoken dialogue systems (Young et al., 2009). A POMDP dialogue manager includes three main parts: a dialogue model representing state information such as the user's goal, the user's dialogue act and the dialogue history; a policy which selects the system's responses based on the inferred dialogue state; and a reward function which specifies the desired behaviour of the system.

In a POMDP system, the dialogue model provides a compact representation for the distribution of the unobserved dialogue state called the *belief state* and it is updated every turn based on the observed user inputs in a process called *belief monitoring*. Exact belief monitoring of the full dialogue state is intractable for all but the simplest systems. One way to address this issue is to represent the state in the compact and approximate form of a dynamic Bayesian Network (BN) (Thomson and Young, 2009). Another way is to maintain probabilities only for the most likely dialogue states. To achieve efficient belief monitoring, indistinguishable states can be grouped into partitions and consequently the belief monitoring is performed on partitions instead of the individual states(Young et al., 2009).

The policy selects the dialogue system's responses (actions) based on the belief state at each turn, and it is typically trained using reinforcement learning with the objective of maximising the expected cumulative reward. The use of reinforcement learning algorithms for POMDP systems usually relies on the observation that a POMDP system can be transformed into a continuous state Markov decision process (MDP) and that the policy optimization problem can then be solved for this newly defined MDP with the guarantee that the solution also optimises the original POMDP (Kaelbling et al., 1998). However, without approximations this is not tractable for any real-world dialogue system. Significant reduction in complexity can be achieved if notions of a summary space and summary actions are introduced. The basic idea is that a successful policy does not need access to all of the information in the belief state and the database, and that summary actions produced by the policy can be mapped back into full actions through a heuristic mapping (Williams and Young, 2005).

The choice of reward function is a dialogue design issue, but it will typically provide positive rewards for satisfying the user's goal, and negative rewards for failure and wasting time. Ideally both the dialogue model and the policy would be designed to maximise the reward function. A typical structure of a POMDP dialogue manager embodying these ideas is depicted in Figure 1.

Adaptivity in the area of spoken dialogue systems can be defined in many ways. For example, Litman and Pan (2002); Chu-Carroll (2000) define adaptivity as ability to change the dialogue management strategy over the course of a dialogue based on user observation and its measure of uncertainty. However, this approach can be shown to be already incorporated in POMDP

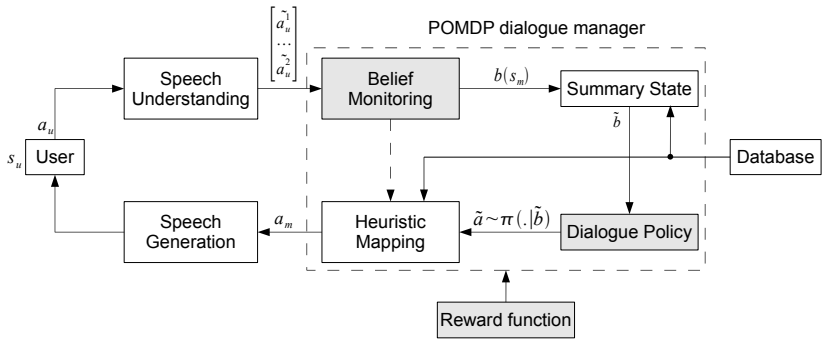


Figure 1: Structure of a POMDP dialogue system: a_u is a user act produced given the user state s_u, $\tilde{a}_u^i$ is an estimate of the users' dialogue acts, $b(s_m)$ is the belief state, $\tilde{b}$ is a summary representation of the belief state, $\tilde{a}$ is a summary action which is later expanded into full dialogue act, a_m, by a heuristic mapping based on the full belief state and the database.

dialogue systems in a very natural way. POMDP dialogue systems explicitly maintain probability over all dialogue states in the form of a belief state; therefore, the policy has access to uncertainty about the estimate of the current dialogue state. This information is then used in the dialogue policies to change its behaviour, e.g. to confirm some information whenever the system is not sure about what was said or to offer when all necessary information is available with high certainty.

Another way is to define adaptivity as the ability to change the dialogue strategy depending on the user being classified as a novice, moderate, or expert (**?**). This method can be easily incorporated into the POMDP dialogue system by adding a variable representing the user type into the dialogue model. Once the dialogue model is extended, the user type is estimated during the course of the conversation as it is done with other parts of the dialogue model. Consequently, the policy takes into account the estimated user type and proposes such system actions that serves the needs of the current users better.

In this report, adaptivity is understood as the ability to learn parameters of a dialogue system from the interaction with real users over a short period of. However, training the parameters of a dialogue system is currently very time consuming task and most of the available algorithms require millions of dialogues (Young et al., 2009; Thomson and Young, 2009). Therefore, training normally takes place in interaction with a simulated user, rather than real users. This raises questions regarding the quality of the approximation as well as the potential discrepancy between simulated and real user behaviour.

As a result, there is a need for efficient fast online learning algorithms that allow the parameters obtained in interaction with the simulated user to be further refined in interaction with real users. Section 2 presents the use of Gaussian processes for fast policy learning which in fact can lead to online policy adaptation with real users. Section 3 describes the Natural Belief Critic algorithm for online dialogue model parameter learning. Section 4 proposes the Kalman Temporal Differences algorithm for management of uncertainty in estimate of the optimal value function.

Another part of this report concerns FT work on the integration of online reinforcement learn-

ing functionality into a handcrafted spoken dialogue system. It is motivated by the following industrial constraint: every system behaviour must be anticipated and controlled. Following the same goal than Singh et al. (2002) and Williams (2008), the objective is to automatically choose online the best alternative among those proposed by the developer in the design tool. This design-centred approach differs from the classical reinforcement learning methods used in the dialogue literature, which make their decisions at the dialogue turn level. Using a novel technique Laroche et al. (2009), the validity of this approach has been demonstrated at multiple occasions Laroche et al. (2010b); Putois et al. (2010). It guarantees a full control of the system by its developers thanks to a unified tool enabling to project the monitoring information in the design view Laroche et al. (2010a). Section 5 concentrates on the description of the proposed technique, while section 6 reports on the evaluation with real users.

The report is concluded in section 7.

2 GP-SARSA: adaptation of a dialogue policy

Gašić et al. (2010) suggested the use of the GP-SARSA algorithm in order to make the learning process faster. The GP-SARSA algorithm approximates the value function by a Gaussian Process. Given the observation of rewards, it estimates the value function utilising its correlations in different parts of the state and the action space defined by the kernel function. There are many kernels that can be used in the approximation of the value function. The main purpose of a kernel is to define a prior knowledge about the value function correlations. Typical kernels are polynomial, parametrised polynomial, and Gaussian. Some of the kernel functions are in a parametrised form, such as Gaussian or parametrised polynomial kernel. These parameters are estimated by an algorithm maximising the marginal likelihood on a corpus of dialogues.

The main advantage of a Gaussian Process approximation lies in efficient use of all the available data points. When this is combined with availability of information about the uncertainty for the current estimate of the policy, it leads to a fast active learning algorithm.

The GP-SARSA was evaluated on a real-world dialogue task showing that this method can learn faster than a grid-based algorithm (Young et al., 2009; Jurčíček et al., 2010a). The results also showed that the variance that a Gaussian Process is estimating can be used to further accelerate policy optimisation.

For more details see the appendix: “Gaussian Processes for Fast Policy Optimisation of POMDP-based Dialogue Managers.”

An interesting by-product of the GP-SARSA algorithm is that it provides an explicit measure of uncertainty at every point in the state space. This could provide a basis for rapid on-line adaptation whereby a set of policies are trained off-line for different user types and then the best policy is selected on-line to minimise the uncertainty. This will be a topic of future work.

3 NBC: adaptation of a dialogue model

A key feature of the POMDP approach is that it includes a specific model of user behaviour. It is important therefore to develop methods by which the parameters can be adapted to specific users.

Jurčíček et al. (2010b) proposed the Natural Belief Critic (NBC) algorithm for learning the dialogue model parameters. The NBC algorithm is based on policy gradient methods (Peters et al., 2005); however, generalisation of these methods was necessary. When using the policy gradient methods, derivatives of the expected reward function with respect to all parameters of the dialogue systems are required. However, this might not be always possible as scalable dialogue systems usually use some form of summary space (Jurčíček et al., 2010a). The difficulty with using summary space is that the mapping from the master space to the summary space depends on the dialogue model parameters through a handcrafted function extracting non-continuous features from the belief state. Consequently, closed form derivatives of the dialogue model parameters are not available. Nevertheless, this problem can be alleviated by assuming that the dialogue model parameters come from a prior distribution that is differentiable with respect to its parameters. In summary, the NBC algorithm estimates the natural gradient of the expected reward based on observed rewards. Then, the resulting gradient is used to adapt the prior distribution of the dialogue model parameters.

The experiments showed that model parameters estimated to maximise the expected reward function result in significantly improved performance compared to the baseline handcrafted parameters. Although the algorithm was evaluated on a user simulator, it is designed for online adaptation of the model parameters in interaction with real users. The method can, for example, utilise rewards provided by real users or automatically measured rewards based on some metrics, e.g. successful completion of a task (uninterrupted call transfer in a call routing application, or a booking of an appointment).

For more details see the appendix: “Natural Belief-Critic: a reinforcement algorithm for parameter estimation in statistical spoken dialogue systems.”

4 Uncertainty management with Kalman Temporal Differences

The Kalman Temporal Differences algorithm Geist and Pietquin (2010) for reinforcement learning has been developed in the framework of task 1.2 and has been reported in deliverable D.1.2. This framework casts the value function approximation into the paradigm of Kalman filtering. The value function is parameterized and Kalman filtering is used to estimate the parameters from the observation of transitions in the state-action space and associated rewards. Kalman filtering computes naturally the uncertainty associated to the estimation of the parameters. This uncertainty can consequently be retro-propagated to the value function itself, traducing the uncertainty one has on the estimation of this function. Since the policy is deducted from the estimated value function, the learnt policy can be very far from optimality in highly uncertain areas of the state-action space. Uncertainty being higher in areas that have been visited less frequently, it is reason-

able to actively try to reduce the uncertainty by forcing the visit of these areas. This is actually related to the well-known exploration vs. exploitation dilemma. Several strategies of exploration can be investigated, based on different usages of the estimated uncertainty, like reported in Geist and Pietquin (2011). In this paper, the results of these different exploration schemes applied to the TownInfo problem are exposed. These results show that choosing adequately the exploration scheme, that is adapting online the policy so as to explore the state-action space, improves the learning rate and enables to learn online a fair policy in few hundreds of dialogue turns.

For more details see the appendix: "Managing Uncertainty within the KTD Framework."

5 Optimising a Handcrafted Dialogue System Design

In the Spoken Dialogue System literature, all studies consider the dialogue move as the unquestionable unit for reinforcement learning. Rather than learning at the dialogue move level, Laroche et al. (2010b) proposes to apply the learning at the design level for three reasons: 1/ to alleviate the high-skill prerequisite for developers, 2/ to reduce the learning complexity by taking into account just the relevant subset of the context and 3/ to have interpretable learning results that carry a reusable usage feedback. Unfortunately, tackling the problem at the design level weakens the Markovian assumptions that are required in most Reinforcement Learning techniques.

Consequently, it was experienced to use a recent non-Markovian algorithm called Compliance Based Reinforcement Learning Laroche et al. (2009). This algorithm tries to identify optimal and suboptimal actions. It learns from optimal actions instead of sub-optimal. The first step of this algorithm consists in the computation of the local compliance of a past system actions to the current optimal policy. This local compliance indicates how this decision deviates from the policy that we consider optimal at that moment. The compliance is computed as a difference between the expected performance implied by the action chosen during the past decisions and the optimal action according to the current policy.

The results show a fast and significant improvement of the system performance of one system misunderstanding less per dialogue in average. For more details see the appendix: "Optimising a Handcrafted Dialogue System Design."

6 CBRL: evaluation on 1013+ appointment scheduling service

Putois et al. (2010) used the same approach as Laroche et al. (2010b). But this time, the algorithm was evaluated on real user of 1013+ appointment scheduling service provided by FT R&D for FT customers in France. When the user calls the system, she/he is presented with an open question asking her/him for the reason of her/his call. If her/his landline is out-of-order, then the SDS performs some automated tests on the line, and if the problem is confirmed, tries and schedules an appointment with the user for a manual intervention. The experiment presented in this article is restricted to the appointment scheduling task, which receives more than 8,000 calls every

month. If the system and the user cannot agree on an appointment slot, the call is transferred to a human operator. The evaluation of real users showed that compliance-based reinforcement learning significantly improved overall performance of the system when compared to the system using the handcrafted setup.

In addition to the support of a large-scale application, Putois et al. (2010) proposes a novel monitoring tool. Thanks to this experience, it was demonstrated that the learning algorithm and the SDS developers are not in conflict. They are two actors working on the same object: the dialogue system. But, they work at a different time scales. The learning algorithm updates its policy after each dialogue while the SDS developers monitor the system behavior more occasionally. The same kind of opposition can be made on their action spaces and their scopes. The learning algorithm is concentrated on the alternative sets and automatic evaluation and ignores the rest, while the SDS developers can apprehend the dialogue application as a whole, as a system or as a service

For more details see the appendix: "Online Reinforcement Learning for Spoken Dialogue Systems: The Story of a Commercial Deployment Success."

7 Conclusions

This report presented research into fast online learning algorithms for the policy and the dialogue model parameters. In this report were proposed several reinforcement learning algorithms. The GP-SARSA algorithm is suitable for fast learning and adaptation of the dialogue policy while the NBC algorithm allows training the dialogue model parameters. Both algorithms are designed to be used in interaction with real users. Whilst both methods still require several hundred dialogues, this is much better than the tens of thousands needed by standard RL algorithms and it opens up the possibility of on-line incremental adaptation in live applications. The KTD algorithm was showed that it actively exploits the information about the uncertainty of the estimate of the value function. Also, the results suggest that this active behaviour translates into improvement of learning rate which enables the policy to be learned online in a few hundreds of dialogue turns. The CBRL algorithm was tested on real users of a real world spoken dialogue system. The results show that a system optimised by the CBRL algorithm has significantly better performance than the baseline system using a carefully tuned handcrafted setup.

References

Chu-Carroll, J. (2000). Mimic: An adaptive mixed initiative spoken dialogue system for information queries. In *ANLP*, pages 97–104.

Gašić, M., Jurčíček, F., Keizer, S., Thomson, B., Yu, K., and Young, S. (2010). Gaussian Processes for Fast Policy Optimisation of POMDP-based Dialogue Managers. In *Interspeech 2010*.

Geist, M. and Pietquin, O. (2010). Kalman temporal differences. *Journal of Artificial Intelligence Research (JAIR)*, 39:489–532.

Geist, M. and Pietquin, O. (2011). Managing Uncertainty within the KTD Framework. In *Proceedings of the Workshop on Active Learning and Experimental Design (AL&E collocated with AISTAT 2010)*, Journal of Machine Learning Research Conference and Workshop Proceedings, Sardinia (Italy). 12 pages - to appear.

Jurčíček, F., Gašić, M., Thomson, B., Lefèvre, F., and Young, S. (2010a). Summary-based policy optimisation. Technical Report D1.4.2, CLASSiC.

Jurčíček, F., Thomson, B., Keizer, S., Gašić, M., Mairesse, F., Yu, K., and Young, S. (2010b). Natural Belief-Critic: a reinforcement algorithm for parameter estimation in statistical spoken dialogue systems. In *Interspeech 2010*.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.

Laroche, R., Bretier, P., and Putois, G. (2010a). Enhanced monitoring tools and online dialogue optimisation merged into a new spoken dialogue system design experience. In *Proceedings of Interspeech*, Chiba (Japan).

Laroche, R., Putois, G., and Bretier, P. (2010b). Optimising a handcrafted dialogue system design. In *Proceedings of Interspeech*, Chiba (Japan).

Laroche, R., Putois, G., Bretier, P., and Bouchon-Meunier, B. (2009). Hybridisation of expertise and reinforcement learning in dialogue systems. In *Proceedings of Interspeech. Special Session: Machine Learning for Adaptivity in Spoken Dialogue*, Brighton (United Knigdom).

Litman, D. J. and Pan, S. (2002). Designing and evaluating an adaptive spoken dialogue system. *User Model. User-Adapt. Interact.*, 12(2-3):111–137.

Peters, J., Vijayakumar, S., and Schaal, S. (2005). Natural actor-critic. In *European Conference on Machine Learning (ECML)*, pages 280–291. Springer.

Putois, G., Laroche, R., and Bretier, P. (2010). Online learning for spoken dialogue systems: The story of the first commercial deployment success. In *Proceedings of SIGDIAL*, Tokyo (Japan).

Singh, S., Litman, D., Kearns, M., and Walker, M. (2002). Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. *Journal of Artificial Intelligence Research*, 16:105–133.

Thomson, B. and Young, S. (2009). Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech and Language*.

Williams, J. (2008). The best of both worlds: Unifying conventional dialog systems and POMDPs. In *Proceedings of the International Conference on Speech and Language Processing (ICSLP)*.

Williams, J. D. and Young, S. (2005). Scaling Up POMDPs for Dialog Management: The Summary POMDP Method. In *IEEE workshop on Automatic Speech Recognition and Understanding (ASRU)*, Cancun, Mexico.

Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., and Yu, K. (2009). The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language*.

A Publications associated with this deliverable

Gaussian Processes for Fast Policy Optimisation of POMDP-based Dialogue Managers

M. Gašić, F. Jurčíček, S. Keizer, F. Mairesse, B. Thomson, K. Yu and S. Young
Cambridge University Engineering Department
Trumpington Street, Cambridge CB2 1PZ, UK
{mg436, fj228, sk561, farm2, brmt2, ky219, sjy}@eng.cam.ac.uk

Abstract

Modelling dialogue as a Partially Observable Markov Decision Process (POMDP) enables a dialogue policy robust to speech understanding errors to be learnt. However, a major challenge in POMDP policy learning is to maintain tractability, so the use of approximation is inevitable. We propose applying Gaussian Processes in Reinforcement learning of optimal POMDP dialogue policies, in order (1) to make the learning process faster and (2) to obtain an estimate of the uncertainty of the approximation. We first demonstrate the idea on a simple voice mail dialogue task and then apply this method to a real-world tourist information dialogue task.

1 Introduction

One of the main challenges in dialogue management is effective handling of speech understanding errors. Instead of hand-crafting the error handler for each dialogue step, statistical approaches allow the optimal dialogue manager behaviour to be learnt automatically. Reinforcement learning (RL), in particular, enables the notion of planning to be embedded in the dialogue management criteria. The objective of the dialogue manager is for each dialogue state to choose such an action that leads to the highest expected long-term reward, which is defined in this framework by the Q-function. This is in contrast to Supervised learning, which estimates a dialogue strategy in such a way as to make it resemble the behaviour from a given corpus, but without directly optimising the overall dialogue success.

Modelling dialogue as a Partially Observable Markov Decision Process (POMDP) allows action selection to be based on the differing levels of uncertainty in each dialogue state as well as the overall reward. This approach requires that a distribution of states (*belief state*) is maintained at each turn. This explicit representation of uncertainty in the POMDP gives it the potential to produce more robust dialogue policies (Young et al., 2010).

The main challenge in the POMDP approach is the tractability of the learning process. A discrete state space POMDP can be perceived as a continuous space MDP where the state space consists of the belief states of the original POMDP. A grid-based approach to policy optimisation assumes discretisation of this space, allowing for discrete space MDP algorithms to be used for learning (Brafman, 1997) and thus approximating the optimal Q-function. Such an approach takes the order of $100,000$ dialogues to train a real-world dialogue manager. Therefore, the training normally takes place in interaction with a simulated user, rather than real users. This raises questions regarding the quality of the approximation as well as the potential discrepancy between simulated and real user behaviour.

Gaussian Processes have been successfully used in Reinforcement learning for continuous space MDPs, for both model-free approaches (Engel et al., 2005) and model-based approaches (Deisenroth et al., 2009). We propose using GP Reinforcement learning in a POMDP dialogue manager to, firstly, speed up the learning process and, secondly, obtain the uncertainty of the approximation. We opt for the model-free approach since it has the potential to allow the policy obtained in interaction with the simulated user to be further refined in interaction with real users.

In the next section, the core idea of the method is explained on a toy dialogue problem where different aspects of GP learning are examined. Following that, in Section 3, it is demonstrated how this methodology can be effectively applied to a real world dialogue. We conclude with Section 4.

2 Gaussian Process RL on a Toy Problem

2.1 Gaussian Process RL

A Gaussian Process is a generative model of Bayesian inference that can be used for function regression (Rasmussen and Williams, 2005). A Gaussian Process is fully defined by a mean and a kernel function. The kernel function defines prior function correlations, which is crucial for obtaining good posterior estimates with just a few observations. GP-Sarsa is an on-line reinforcement learning algorithm for both continuous and discrete MDPs that incorporates GP regression (En-

gel et al., 2005). Given the observation of rewards, it estimates the Q-function utilising its correlations in different parts of the state and the action space defined by the kernel function. It also gives a variance of the estimate, thus modelling the uncertainty of the approximation.

2.2 Voice Mail Dialogue Task

In order to demonstrate how this methodology can be applied to a dialogue system, we first explain the idea on the voice mail dialogue problem (Williams, 2006).

The state space of this task consists of three states: the user asked for the message either to be saved or deleted, or the dialogue ended. The system can take three actions: ask the user what to do, save or delete the message. The observation of what the user wants is corrupted with noise, therefore we model this as a three-state POMDP. This POMDP can be viewed a continuous MDP, where the MDP state is the POMDP belief state, a 3-dimensional vector of probabilities. For both learning and evaluation, a simulated user is used which makes a mistake with probability 0.3 and terminates the dialogue after at most 10 turns. In the final state, it gives a positive reward of 10 or a penalty of -100 depending on whether the system performed a correct action or not. Each intermediate state receives the penalty of -1. In order to keep the problem simple, a model defining transition and observation probabilities is assumed so that the belief can be easily updated, but the policy optimisation is performed in an on-line fashion.

2.3 Kernel Choice for GP-Sarsa

The choice of kernel function is very important since it defines the prior knowledge about the Q-function correlations. They have to be defined on both states and actions. In the voice mail dialogue problem the action space is discrete, so we opt for a simple δ kernel over actions:

$$k(a, a') = 1 - \delta_a(a'), \qquad (1)$$

where δ_a is the Kronecker delta function. The state space is a 3-dimensional continuous space and the kernel functions over the state space that we explore are given in Tab 2.3. Each kernel function defines a different correlation. The polynomial kernel views elements of the state vector as features, the dot-product of which defines the correlation. They can be given different relevance r_i in the parametrised version. The Gaussian kernel accounts for smoothness, *i.e.*, if two states are close to each other the Q-function in these states is correlated. The scaled norm kernel defines positive correlations in the points that are close to each other and a negative correlation otherwise. This is particularly useful for the voice mail problem, where, if two belief states are very different, taking the same action in these states generates a negatively correlated reward.

kernel function	expression
polynomial	$k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$
parametrised poly.	$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{D} \frac{x_i x_i'}{r_i^2}$
Gaussian	$k(\mathbf{x}, \mathbf{x}') = p^2 \exp \frac{-\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma_k^2}$
scaled norm	$k(\mathbf{x}, \mathbf{x}') = 1 - \frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\|\mathbf{x}\|^2 \|\mathbf{x}'\|^2}$

Table 1: Kernel functions

2.4 Optimisation of Kernel Parameters

Some kernel functions are in a parametrised form, such as Gaussian or parametrised polynomial kernel. These parameters, also called *the hyper-parameters*, are estimated by maximising the marginal likelihood[1] on a given corpus (Rasmussen and Williams, 2005). We adapted the avaliable code (Rasmussen and Williams, 2005) for the Reinforcement learning framework to obtain the optimal hyper-parameters using a dialogue corpus labelled with states, actions and rewards.

2.5 Grid-based RL Algorithms

To assess the performance of GP-Sarsa, it was compared with a standard grid-based algorithm used in (Young et al., 2010). The grid-based approach discretises the continuous space into regions with their representative points. This then allows discrete MDP algorithms to be used for policy optimisation, in this case the Monte Carlo Control (MCC) algorithm (Sutton and Barto, 1998).

2.6 Optimal POMDP Policy

The optimal POMDP policy was obtained using the POMDP solver toolkit (Cassandra, 2005), which implements the Point Based Value Iteration algorithm to solve the POMDP off-line using the underlying transition and observation probabilities. We used 300 sample dialogues between the dialogue manager governed by this policy and the simulated user as data for optimisation of the kernel hyper-parameters (see Section 2.4).

2.7 Training set-up and Evaluation

The dialogue manager was trained in interaction with the simulated user and the performance was compared between the grid-based MCC algorithm and GP-Sarsa across different kernel functions from Table 2.3.

The intention was, not only to test which algorithm yields the best policy performance, but also to examine the speed of convergence to the optimal policy. All the algorithms use an ϵ-greedy approach where the exploration rate ϵ was fixed at 0.1. The learning process greatly depends on

[1] Also called *evidence maximisation* in the literature.

the actions that are taken during exploration. If early on during the training, the systems discovers a path that generates high rewards due to a lucky choice of actions, then the convergence is faster. Therefore, to alleviate this, we adopt the following procedure. For every training set-up, exactly the same training iterations were performed using 1000 different random generator seedings. After every 20 dialogues the resulting 1000 partially optimised policies were evaluated. Each of them was tested on 1000 dialogues. The average reward of these 1000 dialogues provides just one point in Fig. 1.

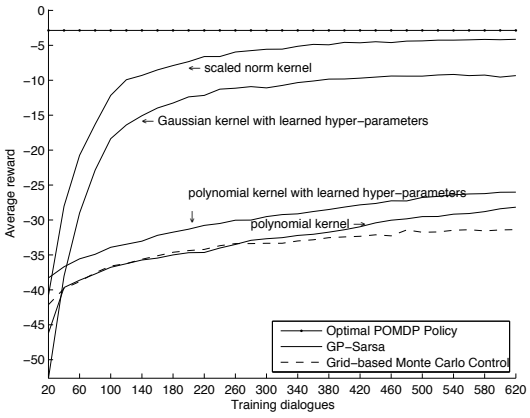


Figure 1: Evaluation results on Voice Mail task

The grid-based MCC algorithm used a Euclidian distance to generate the grid by adding every point that was further than 0.01 from other points as a representative of a new region. As can be seen from Fig 1, the grid-Based MCC algorithm has a relatively slow convergence rate. GP-Sarsa with the polynomial kernel exhibited a learning rate similar to MCC in the first 300 training dialogues, continuing with a more upward learning trend. The parametrised polynomial kernel performs slightly better. The Gaussian kernel, however, achieves a much faster learning rate. The scaled norm kernel achieved close to optimal performance in 400 dialogues, with a much higher convergence rate then the other methods.

3 Gaussian Process RL on a Real-world Task

3.1 HIS Dialogue Manager on CamInfo Domain

We investigate the use of GP-Sarsa in a real-world task by extending the Hidden Information State (HIS) dialogue manager (Young et al., 2010). The application domain is tourist information for Cambridge, whereby the user can ask for information about a restaurant, hotel, museum or another tourist attraction in the local area. The database consists of more than 400 entities each of which has up to 10 attributes that the user can query.
The HIS dialogue manager is a POMDP-based dialogue manager that can tractably maintain belief states for large domains. The key feature of this approach is the grouping of possible user goals into *partitions*, using relationships between different attributes from possible user goals. Partitions are combined with possible user dialogue actions from the N-best user input as well as with the dialogue history. This combination forms the state space – the set of *hypotheses*, the probability distribution over which is maintained during the dialogue. Since the number of states for any real-world problem is too large, for tractable policy learning, both the state and the action space are mapped into smaller scale summary spaces. Once an adequate summary action is found in the summary space, it is mapped back to form an action in the original *master space*.

3.2 Kernel Choice for GP-Sarsa

The summary state in the HIS system is a four-dimensional space consisting of two elements that are continuous (the probability of the top two hypotheses) and two discrete elements (one relating the portion of the database entries that matches the top partition and the other relating to the last user action type). The summary action space is discrete and consists of eleven elements.
In order to apply the GP-Sarsa algorithm, a kernel function needs to be specified for both the summary state space and the summary action space. The nature of this space is quite different from the one described in the toy problem. Therefore, applying a kernel that has negative correlations, such as the scaled norm kernel (Table 2.3) might give unexpected results. More specifically, for a given summary action, the mapping procedure finds the most appropriate action to perform if such an action exists. This can lead to a lower reward if the summary action is not adequate but would rarely lead to negatively correlated rewards. Also, parametrised kernels could not be used for this task, since there was no corpus available for hyper-parameter optimisation. The polynomial kernel (Table 2.3) assumes that the elements of the space are features. Due to the way the probability is maintained over this very large state space, the continuous variables potentially encode more information than in the simple toy problem. Therefore, we used the polynomial kernel for the continuous elements. For discrete elements, we utilise the δ-kernel (Eq. 2.3).

3.3 Active Learning GP-Sarsa

The GP RL framework enables modelling the uncertainty of the approximation. The uncertainty estimate can be used to decide which actions

to take during the exploration (Deisenroth et al., 2009). In detail, instead of a random action, the action in which the Q-function for the current state has the highest variance is taken.

3.4 Training Set-up and Evaluation

Policy optimisation is performed by interacting with a simulated user on the dialogue act level. The simulated user gives a reward at the final state of the dialogue, and that is 20 if the dialogue was successful, 0 otherwise, less the number of turns taken to fulfil the user goal. The simulated user takes a maximum of 100 turns in each dialogue, terminating it when all the necessary information has been obtained or if it looses patience.
A grid-based MCC algorithm provides the baseline method. The distance metric used ensures that the number of regions in the grid is small enough for the learning to be tractable (Young et al., 2010).
In order to measure how fast each algorithm learns, a similar training set-up to the one presented in Section 2.7 was adopted and the averaged results are plotted on the graph, Fig. 2.

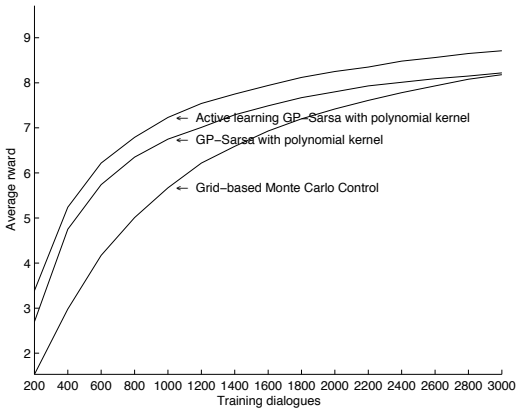


Figure 2: Evaluation results on CamInfo task

The results show that in the very early stage of learning, *i.e.*, during the first 400 dialogues, the GP-based method learns faster. Also, the learning process can be accelerated by adopting the active learning framework where the actions are selected based on the estimated uncertainty.
After performing many iterations in an incremental noise learning set-up (Young et al., 2010) both the GP-Sarsa and the grid-based MCC algorithms converge to the same performance.

4 Conclusions

This paper has described how Gaussian Processes in Reinforcement learning can be successfully applied to dialogue management. We implemented a GP-Sarsa algorithm on a toy dialogue problem, showing that with an appropriate kernel function faster convergence can be achieved. We also demonstrated how kernel parameters can be learnt from a dialogue corpus, thus creating a bridge between Supervised and Reinforcement learning methods in dialogue management. We applied GP-Sarsa to a real-world dialogue task showing that, on average, this method can learn faster than a grid-based algorithm. We also showed that the variance that GP is estimating can be used in an Active learning setting to further accelerate policy optimisation.
Further research is needed in the area of kernel function selection. The results here suggest that the GP framework can facilitate faster learning, which potentially allows the use of larger summary spaces. In addition, being able to learn efficiently from a small number of dialogues offers the potential for learning from direct interaction with real users.

Acknowledgements

The authors would like to thank Carl Rasmussen for valuable discussions. This research was partly funded by the UK EPSRC under grant agreement EP/F013930/1 and by the EU FP7 Programme under grant agreement 216594 (CLASSiC project).

References

RI Brafman. 1997. A Heuristic Variable Grid Solution Method for POMDPs. In *AAAI*, Cambridge, MA.

AR Cassandra. 2005. POMDP solver. `http://www.cassandra.org/pomdp/code/index.shtml`.

MP Deisenroth, CE Rasmussen, and J Peters. 2009. Gaussian Process Dynamic Programming. *Neurocomput.*, 72(7-9):1508–1524.

Y Engel, S Mannor, and R Meir. 2005. Reinforcement learning with Gaussian processes. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 201–208, New York, NY.

CE Rasmussen and CKI Williams. 2005. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA.

RS Sutton and AG Barto. 1998. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA.

JD Williams. 2006. *Partially Observable Markov Decision Processes for Spoken Dialogue Management*. Ph.D. thesis, University of Cambridge.

SJ Young, M Gašić, S Keizer, F Mairesse, J Schatzmann, B Thomson, and K Yu. 2010. The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language*, 24(2):150–174.

Natural Belief-Critic: a reinforcement algorithm for parameter estimation in statistical spoken dialogue systems

F. Jurčíček, B. Thomson, S. Keizer, F. Mairesse, M. Gašić, K. Yu, and S. Young

Engineering Department, Cambridge University, CB2 1PZ, UK

{fj228, brmt2, sk561, f.mairesse, mg436, ky219, sjy}@eng.cam.ac.uk

Abstract

This paper presents a novel algorithm for learning parameters in statistical dialogue systems which are modelled as Partially Observable Markov Decision Processes (POMDPs). The three main components of a POMDP dialogue manager are a dialogue model representing dialogue state information; a policy which selects the system's responses based on the inferred state; and a reward function which specifies the desired behaviour of the system. Ideally both the model parameters and the policy would be designed to maximise the reward function. However, whilst there are many techniques available for learning the optimal policy, there are no good ways of learning the optimal model parameters that scale to real-world dialogue systems.

The Natural Belief-Critic (NBC) algorithm presented in this paper is a policy gradient method which offers a solution to this problem. Based on observed rewards, the algorithm estimates the natural gradient of the expected reward. The resulting gradient is then used to adapt the prior distribution of the dialogue model parameters. The algorithm is evaluated on a spoken dialogue system in the tourist information domain. The experiments show that model parameters estimated to maximise the reward function result in significantly improved performance compared to the baseline handcrafted parameters.

Index Terms: spoken dialogue systems, reinforcement learning, POMDP, dialogue management

1. Introduction

A POMDP dialogue manager includes three main parts: a dialogue model representing state information such as the user's goal, the user's dialogue act and the dialogue history; a policy which selects the system's responses based on the inferred dialogue state; and a reward function which specifies the desired behaviour of the system. In a POMDP system, the dialogue model provides a compact representation for the distribution of the unobserved dialogue state called the *belief state* and it is updated every turn based on the observed user inputs in a process called *belief monitoring*. Exact belief monitoring of the full dialogue state is intractable for all but the simplest systems. However, if the state is represented in the compact and approximate form of a dynamic Bayesian Network (BN), factored according to the slots in the system then by exploiting the conditional independence of the network nodes, a tractable system can be built [1]. In this case, the parameters of the model are the conditional distributions describing the nodes in the network.

The policy selects the dialogue system's responses (actions) based on the belief state at each turn, and it is typically trained using reinforcement learning with the objective of maximising the reward function. While there are many efficient techniques for learning the policy parameters [2, 3, 4], there are no good ways of learning the model parameters which scale to real-world dialogue systems. Hence, in virtually all current systems, the dialogue model parameters are handcrafted by a system designer [1, 3]. Ideally, one would like to estimate the parameters from the interactions with the user and some attempts have been made in this direction. For example, maximum likelihood estimates can be obtained by annotating the correct dialogue state in a corpus of real dialogues. However, in many real dialogues, some components of the dialogue state, especially the user's goal, are hard to determine. Hence, in practice this approach is restricted to cases where the user's goal remains constant and the dialogue is simple to annotate [5]. An alternative is to use algorithms such as Expectation-Maximization [6] or Expectation-Propagation [7] which can infer hidden state information. However, again these algorithms usually require the user goal to remain constant and even then it is not clear to what extent likelihood maximisation over a dialogue corpus correlates with the expected reward of the dialogue system.

This paper presents a novel reinforcement algorithm called Natural Belief-Critic (NBC) for learning the parameters of a dialogue model which maximise the reward function. The method is presented and evaluated in the context of the BUDS POMDP dialogue manager which uses a dynamic Bayesian Network to represent the dialogue state. However, the method is sufficiently general that it could be used to optimise virtually any parameterised dialogue model. Furthermore, unlike most of the maximum likelihood methods used so far, the NBC algorithm does not require that the user goal remains constant.

The paper is structured as follows. Section 2 briefly describes the BUDS dialogue manager and the method it uses for policy representation [1]. Section 3 then describes policy gradients methods and a specific form called the Natural Actor-Critic (NAC) algorithm which is used to optimise the BUDS policy. In Section 4, the proposed Natural Belief-Critic algorithm is presented as a generalisation of the NAC algorithm and then in Section 5 it is evaluated on a system designed for the tourist information domain. Finally, Section 6 presents conclusions.

2. BUDS dialogue manager

In a POMDP dialogue system, the true dialogue state s_t is unknown. Therefore, the policy selects an action a_t at time t based on the distribution over all states called the belief state, $b(s_t)$. The estimate of the belief state depends on past observations and actions. If the system is Markovian then the belief state b_t depends only on the previous belief state b_{t-1}, the current observation o_t and the last system action a_{t-1}:

$$b(s_t;\tau) = k \cdot p(o_t|s_t;\tau) \sum_{s_{t-1}} p(s_t|a_{t-1}, s_{t-1};\tau) b(s_{t-1}|h_{t-1};\tau) \quad (1)$$

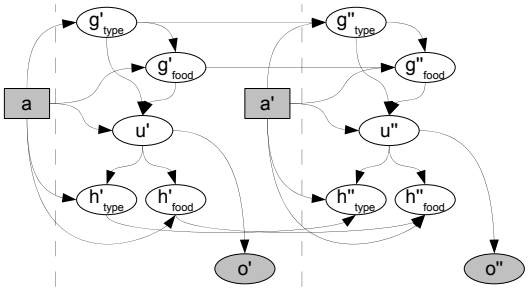


Figure 1: An example factorisation for the Bayesian network representing part of a tourist information dialogue system.

where the transition probability function $p(s_t|a_{t-1}, s_{t-1}; \tau)$ and the observation probability $p(o_t|s_t; \tau)$ represent the dialogue model which is parameterised by τ and k is a normalisation constant.

2.1. The dialogue model

A naive implementation of (1) is not tractable since there are billions of states in a real-world spoken dialogue system[1]. Thus, the BUDS dialogue manager uses a Bayesian Network (BN) to represent the state of the POMDP system, where the network is factored according to the slots in the system [1]. Provided that each slot or network node has only a few dependencies, tractable systems can be built and belief estimates maintained with acceptable accuracy using approximate inference [8].

The BUDS dialogue state is factored into three components: the user goal g, the user action u and the dialogue history h. In addition, the goal and the history are further factored into sub-goals g_i and sub-histories h_i according to a set of slots, $i \in \mathcal{I}$, in the system. For example, in a tourist information system typical sub-goals might be the type of venue required ("type") or the type of food ("food"). The sub-history nodes allow the system designer to store information about whether a user requested information or the system informed the user about some slot. The user action u is the estimate of the true dialogue act from the observation o.[2] Fig. 1 shows the resulting network for two time-slices of a two-slot system based on this idea.

The BN model parameters τ comprise the set of conditional probabilities of the node values. For example, the "food" sub-goal values are described by the probability $p(g''_{food}|g'_{food}, g''_{type}, a'; \tau_{food})$ parameterised by τ_{food}. To reduce the number of parameters specifying the distributions in the sub-goals, some parameters are tied together on the assumption that the probability of change in the sub-goals is constant given the last system action and the parent sub-goal. For example, the probability of change from "Chinese" to "Indian" in the sub-goal "food" is equal to the probability of change from "Chinese" to "Italian".

2.2. The Policy

The BUDS dialogue manager uses a stochastic policy $\pi(a|b; \theta)$ which gives the probability of taking action a given belief state b and policy parameters θ. When used in the dialogue manager, the policy distribution is sampled to yield the required action at each turn. To reduce complexity, for every action a, the belief state is mapped into a vector of features, $\Phi_a(b)$ and the policy is then approximated by a softmax function:

$$\pi(a_t|b(\cdot;\tau)); \theta) \approx \frac{e^{\theta^T \cdot \Phi_{a_t}(b(\cdot;\tau))}}{\sum_{\tilde{a}} e^{\theta^T \cdot \Phi_{\tilde{a}}(b(\cdot;\tau))}}. \quad (2)$$

To estimate the policy parameters, BUDS uses the Natural Actor-Critic (NAC) algorithm [4] (see Section 3).

A further reduction in complexity can be achieved by utilising summary actions [1]. For example, if the dialogue manager confirms the value of some sub-goal then it should always confirm the most likely value. As a result, the full set of actions is not needed. The mapping of the summary actions into full dialogue acts is performed by a handcrafted function based on the information in the belief state.

There are a variety of possible forms for the Φ function [2]. The BUDS dialogue manager uses factored grid-based approximation. In this case, for every node in the BN a set of binary features is generated based on the probabilities of two most likely values. BUDS also supports handcrafted policies which are designed by an expert. These policies deterministically choose which action to take given the features.

3. Policy gradients

The objective of reinforcement learning is to find a policy π which maximises the expected reward $J(\theta)$:

$$J(\theta) = E\Big[\frac{1}{T}\sum_{t=1}^{T} r(s_t, a_t) \mid \pi_\theta\Big],$$

where $r(s_t, a_t)$ is the reward when taking action a_t in state s_t.

Learning θ can be achieved by a gradient ascent which iteratively adds a multiple of the gradient to the parameters being estimated. Using "the log likelihood-ratio trick" and Monte Carlo sampling, the gradient can be estimated as follows:

$$\nabla J(\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} \nabla \log \pi(a_t^n|b_t^n; \theta) R_n \quad (3)$$

where the sampled dialogues are numbered $n = 1, \ldots, N$, the n-th dialogue has a length of T_n turns, and $R_n = \frac{1}{T_n}\sum_{t=1}^{T_n} r(s_t, a_t)$ is the reward accumulated in dialogue n. To obtain a closed form solution for the gradient ∇J, the policy π must be differentiable w.r.t. θ. Conveniently, the softmax function in (2) is "linear" w.r.t. the parameters θ. Thus, it is easy to derive an analytic form for the gradient ∇J.

Although (3) can provide an estimate for the "vanilla" gradient, it has been shown that the natural gradient $\widetilde{\nabla} J(\theta) = F_\theta^{-1} \nabla J(\theta)$ is more effective for optimisation of statistical models where F_θ is the Fisher Information Matrix [9]. Based on this idea, Peters et al. developed the Natural Actor-Critic (NAC) algorithm which estimates a *natural gradient* of the expected reward function [4]. The appealing part of the NAC algorithm is that in practice the Fisher Information Matrix does not need to be explicitly computed. To obtain the natural gradient, w, of $J(\theta)$, NAC uses a least square method to solve the following set of equations:

$$R_n = \left[\sum_{t=1}^{T_n} \nabla \log \pi(a_t^n|b_t^n; \theta)^T\right] \cdot w + C \quad \forall n \in \{1, \ldots, N\}.$$

Once w has been found, the policy parameters can be iteratively improved by $\theta' \leftarrow \theta + \beta w$, where β is a step size.

Of all the policy optimisation algorithms tested with BUDS, the NAC algorithm has proved to be the most robust suggesting that the use of the natural gradient is critical. The question

[1] Note that if a dialogue system has 10 slots and each slot has 10 different values then there are 10^{10} distinct states.

[2] In the BUDS dialogue manager, the observations and system actions are implemented as dialogue acts. A dialogue act conveys the user or system intention (such as inform, request, etc) and a list of slot-value pairs (e.g. type=hotel, area=east).

therefore arises whether this type of policy gradient method can be generalised to optimise not just the policy but the parameters of the dialogue model as well.

4. Natural Belief-Critic algorithm

The difficulty with using policy gradient methods for learning the parameters of the dialogue model is that since the function Φ, which extracts features from the belief state, is usually a handcrafted function of non-continuous features, the policy is not usually differentiable w.r.t. τ. However, this problem can be alleviated by assuming that the model parameters τ come from a prior distribution $p(\tau;\alpha)$ that is differentiable w.r.t. the parameters α. This leads to a generalisation of the NAC algorithm called the Natural Belief-Critic (NBC) algorithm.

The goal of NBC is to learn the parameters α of the prior distribution while maximising the expected reward. The algorithm assumes that the policy is fixed during training. At each iteration, the NBC algorithm samples the model parameters, executes dialogues, and stores the rewards observed at the end of each dialogue. After collecting sufficient statistics, the algorithm updates the prior distribution based on the observed rewards. Finally, the expected values for τ given the distribution $p(\tau;\alpha)$ provide the new estimates for τ.

The techniques used in NAC to compute the natural gradient can be extended to the NBC algorithm since both algorithms sample from the distribution for which they are learning the parameters. The only difference is that NBC samples only at the beginning of a dialogue. As a result, NBC solves the following set of equations:

$$R_n = \nabla \log p(\tau^n;\alpha)^T \cdot w + C \quad \forall n \in \{1,\ldots,N\} \qquad (4)$$

to obtain the natural gradient w of the expected reward.

In order to use NBC in practice a prior for the model parameters τ is needed. Since the parameters of the BN described in Section 2.1 are parameters of multiple multinomial distributions, a product of Dirichlet distributions provides a convenient prior.

Formally, for every node $j \in \{1,\ldots,J\}$ in the BN, there are parameters τ_j describing a probability $p(j|par(j);\tau_j)$ where the function $par(j)$ defines the parents of the node j. Let $|par(j)|$ be the number of distinct combinations of values of the parents of j. Then, τ_j is composed of parameters of $|par(j)|$ multinomial distributions and it is structured as follows: $\tau_j = \left[\tau_{j,1},\ldots,\tau_{j,|par(j)|}\right]$. Consequently, a prior for τ_j can be formed from a product of Dirichlet distributions: $\prod_{k=1}^{|par(j)|} Dir(\tau_{j,k};\alpha_{j,k})$ parameterised by $\alpha_{j,k}$. Let the vector $\tau = [\tau_1,\ldots,\tau_J]$ be a vector of all parameters in the BN. Then, the probability $p(\tau;\alpha)$ from (4) can be defined as $p(\tau;\alpha) = \prod_{j=1}^{J}\prod_{k=1}^{|par(j)|} Dir(\tau_{j,k};\alpha_{j,k})$ which has a closed form log-derivative w.r.t. α and can be used in (4) to compute the natural gradient w. The complete NBC algorithm is described in Algorithm 1.

Algorithm 1 Natural Belief-Critic

Let τ be the parameters of the dialogue model
Let $p(\tau;\alpha)$ be a prior for τ parameterised by α
Let α_1 be the initial parameters of the prior for τ
Let π be a fixed policy
Let N be the number of dialogues sampled in each iteration
Let M be the number of training iterations
Let β be a step size

for $i = 1$ to M **do**
  `Collecting statistics:`
  for $n = 1$ to N **do**
    Draw parameters $\tau^n \sim p(\tau^n;\alpha_i)$
    Execute the dialogue according the policy π
    Observe the reward R_n
  end for
  `Critic evaluation:`
  Choose w_i to minimize the sum of the squares of the errors of $R_n = \nabla \log p(\tau^n;\alpha)^T \cdot w_i + C$
  `Parameter update:`
  $\alpha_{i+1} \leftarrow \alpha_i + \beta w_i$
end for

5. Evaluation

An experimental evaluation of the Natural Belief-Critic algorithm was conducted using the BUDS dialogue system described in Section 2. The goal of the evaluation was to test whether the NBC algorithm could improve on a set of carefully handcrafted model parameters which had been refined over time to optimise performance. The evaluation was in two parts. Firstly a set of model parameters were estimated using a finely tuned handcrafted policy, and secondly, a set of model parameters were estimated using a stochastic policy trained using the NAC algorithm. In both cases, the results were compared to the performance obtained using the initial handcrafted model parameters.

The systems were trained and tested using an agenda based user simulator, for the Town-Info domain which provides tourist information for an imaginary town [1, 3]. The user simulator incorporates a semantic concept confusion model, which enables the systems to be trained and tested across a range of semantic error rates. The reward function used in all experiments awards 100 minus the number of dialogue turns for a successful dialogue and 0 minus the number of turns for an unsuccessful one.

5.1. Dialogue model for the Town-Info domain

The Bayesian Network for the Town-Info domain contains nine sub-goals: name of the venue, type of venue, area, price range, nearness to a particular location, type of drinks, food type, number of stars and type of music. Every sub-goal has a corresponding sub-history node. The network also has nodes to represent address, telephone number, a comment on the venue and the price. However, for these only their sub-history nodes are used since a user can only ask for values of these slots and cannot specify them as query constraints. Finally, the network has two special nodes. The "method" node stores the probability that the user is searching for a venue by constraint rather than by name. The "discourse" node infers whether a user wants the system to repeat the last system action, restart the dialogue, end the dialogue or provide the user with more information about the last offered venue. Although the dialogue manager does not ask about these nodes explicitly, their values are inferred just like any other node.

The history, "method", and "discourse" nodes use fully parameterised conditional probabilities in order to capture the detailed characteristics of dialogue flow. All of the other sub-goal nodes use parameter tying as described in Section 2.1. Overall this results in a total of 577 parameters in the dialogue model.

5.2. Experiments

Dialogue model parameters using the handcrafted policy were estimated by running the NBC algorithm for 50 iterations with the simulator set to give a 40% error rate. In each iteration, 16k dialogues were sampled. Both the baseline system and the system with the learnt BN parameters were evaluated over error rates ranging from 0% to 50%. At each error rate, 5000 dia-

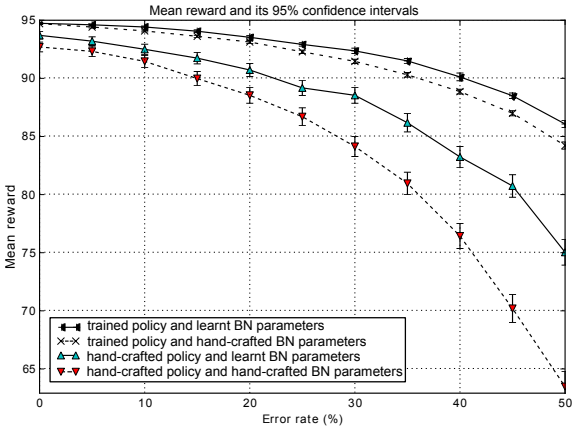


Figure 2: Comparison of the mean rewards of the handcrafted BN model parameters and the parameters learnt by NBC when trained using both a handcrafted policy and a trained policy.

logues were simulated and to reduce the variance of results, this training and evaluation procedure was executed 5 times. The averaged results along with 95% confidence intervals are depicted in Fig. 2. As can be seen, the system with trained BN parameters significantly outperforms the system with handcrafted parameters especially at high error rates. For example, at 40% error rate, the mean reward was increased by 8.3% ($p < 0.05$). Inspection of the results suggests that this improvement can be mostly attributed to the sub-optimality of the handcrafted policy and the ability of the learnt BN parameters compensate for this.

In the second experiment, the NBC algorithm was used to estimate a set of model parameters for a system using an optimised stochastic policy. The full training procedure was executed in three steps. First, a stochastic policy was learnt using a dialogue model initialised with handcrafted parameters. To train the policy, the NAC algorithm was executed for 200 iterations at a 40% error rate and in each iteration 4000 dialogues were simulated. Second, NBC was used to train BN parameters using the newly trained stochastic policy. Thirdly, the policy was retrained using NAC to take advantage of the improved model parameters. The final system was evaluated as in the first task; although in this case, the training and evaluation procedure was executed 20 times. The results, depicted in Fig. 2, show that a system with model parameters trained using NBC significantly improves on the system with handcrafted model parameters even when used with a trained policy. At 40% error rate, the mean reward was increased by 2.2% ($p < 0.05$). Further iterations of model parameter estimation and policy optimization did not lead to any further improvement in performance.

Inspection of the learnt model parameters compared to the handcrafted parameters based on KL-divergence showed that greatest effect of the NBC-based optimisation was on the "method" and "discourse" nodes. This is in line with expectations since the probabilities of change in these nodes are less intuitive and they are therefore much harder to set manually.

The NBC algorithm was also tested with initial model parameters different to the handcrafted ones. Simulations showed that although the algorithm is able to improve on arbitrary initialisations, the maximum performance achieved is sensitive to the initialisation, presumably because the algorithm converges to differing local optima.

Experiments were also conducted with uninformative (uniform) priors on the model parameters; though, they were not entirely successful since in this case, the final rewards were lower by 10%-20% in comparison with the rewards obtained when using the handcrafted parameters. It appears that the NBC algorithm too quickly reduces the variance of the prior distribution. Consequently, it limits exploration of the dialogue model parameters.

The NBC algorithm can also be understood as a random search algorithm. Thus, other state-of-the-art random search techniques such as SPSA [10] and CMA-ES [11] can be used. However, informal testing with these techniques yielded no further improvement to the results reported here.

6. Conclusion

This paper has proposed a novel method called the Natural Belief Critic algorithm for estimating the model parameters of a POMDP-based dialogue system so as to maximise the reward. Based on observed rewards obtained in a set of training dialogues, the algorithm estimates the natural gradient of the expected reward of a dialogue system and then adapts the Dirichlet prior distributions of the model parameters. Simulations have shown that the NBC algorithm significantly improves upon an initial set of handcrafted model parameters when used with both handcrafted and trained policies. Although the NBC algorithm converges reliably, the achievable maximum reward is sensitive to the initialisation. Thus the algorithm is most effective for improving on an existing set of model parameters which have either been handcrafted or estimated by other methods such as maximum likelihood.

Acknowledgment: This research was partly funded by the UK EPSRC under grant agreement EP/F013930/1 and by the EU FP7 Programme under grant agreement 216594 (CLASSIC project: www.classic-project.org).

7. References

[1] B. Thomson and S. Young, "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems." *Computer Speech and Language*, vol. 24, no. 4, pp. 562 – 588, 2010.

[2] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, ser. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 1998.

[3] S. Young, M. Gašić, 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 and Language, In press*, 2009.

[4] J. Peters, S. Vijayakumar, and S. Schaal, "Natural Actor-Critic," in *European Conference on Machine Learning (ECML)*. Springer, 2005, pp. 280–291.

[5] D. Kim, H. S. Sim, K. Kim, J. H. Kim, H. Kim, and J. W. Sung, "Effects of User Modeling on POMDP-based Dialogue Systems," in *Proceedings of Interspeech*, 2008.

[6] U. Syed and J. D. Williams, "Using automatically transcribed dialogs to learn user models in a spoken dialog system," in *HLT*, Morristown, USA, 2008, pp. 121–124.

[7] B. Thomson, "Statistical methods for spoken dialogue management," Ph.D. dissertation, University of Cambridge, 2010.

[8] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[9] S. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998.

[10] J. C. Spall, *Introduction to Stochastic Search and Optimization*. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[11] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

JMLR: Workshop and Conference Proceedings 10 (2010) 1–12 Workshop on Active Learning and Experimental Design

Managing Uncertainty within the KTD Framework

Matthieu Geist MATTHIEU.GEIST@SUPELEC.FR
Olivier Pietquin OLIVIER.PIETQUIN@SUPELEC.FR
IMS Research Group, Supélec, Metz, France

Editor: I. Guyon, G. Cawley, G. Dror, V. Lemaire, and A. Statnikov

Abstract

The dilemma between exploration and exploitation is an important topic in reinforcement learning (RL). Most successful approaches in addressing this problem tend to use some uncertainty information about values estimated during learning. On another hand, scalability is known as being a lack of RL algorithms and value function approximation has become a major topic of research. Both problems arise in real-world applications, however few approaches allow approximating the value function while maintaining uncertainty information about estimates. Even fewer use this information in the purpose of addressing the exploration/exploitation dilemma. In this paper, we show how such an uncertainty information can be derived from a Kalman-based Temporal Differences (KTD) framework and how it can be used.

Keywords: Value function approximation, active learning, exploration/exploitation dilemma

1. Introduction

Reinforcement learning (RL) (Sutton and Barto, 1996) is the machine learning answer to the well-known problem of optimal control of dynamic systems. In this paradigm, an agent learns to control its *environment* (*i.e.*, the dynamic system) through examples of actual interactions. To each of these interactions is associated an immediate reward which is a local hint about the quality of the current control policy. More formally, at each (discrete) time step i the dynamic system to be controlled is in a state s_i. The agent chooses an action a_i, and the dynamic system is then driven in a new state, say s_{i+1}, following its own dynamics. The agent receives a reward r_i associated to the transition (s_i, a_i, s_{i+1}). The agent's objective is to maximize the expected cumulative rewards, which it internally models as a so-called value or Q-function (see later). In the most challenging cases, learning has to be done online and the agent has to control the system while trying to learn the optimal policy. A major issue is then the choice of the behavior policy and the associated dilemma between exploration and exploitation (which can be linked to active learning). Indeed at each time step, the agent can choose an optimal action according to its (maybe) imperfect knowledge of the environment (exploitation) or an action considered to be suboptimal so as to improve its knowledge (exploration) and subsequently its policy. The ϵ-greedy action selection is a popular choice which consists in selecting the greedy action with probability $1-\epsilon$, and an equally distributed random action with probability ϵ. Another popular scheme is the *softmax* action selection (Sutton and Barto, 1996) drawing the behavior action from a Gibbs distribution. Most successful approaches tend to use an uncertainty information

to choose between exploration and exploitation but also to drive exploration. Dearden et al. (1998) maintain a distribution for each Q-value. They propose two schemes. The first one consists in sampling the action according to the Q-value distribution. The second one uses a myopic value of imperfect information which approximates the utility of an information-gathering action in terms of the expected improvement of the decision quality. Strehl and Littman (2006) maintain a confidence interval for each Q-value and the policy is greedy respectively to the upper bound of this interval. This approach allows deriving probably-approximately-correct (PAC) bounds. Sakaguchi and Takano (2004) use a Gibbs policy. However a reliability index (actually a form of uncertainty) is used instead of the more classic temperature parameter. Most of these approaches are designed for problems where an exact (tabular) representation of the value function is possible. Nevertheless, approximating the value in the case of large state spaces is another topic of importance in RL. There are some model-based algorithms which address this problem (Kakade et al., 2003; Jong and Stone, 2007; Li et al., 2009b). They imply approximating the model in addition to the value function. However we focus here an pure model-free approaches (just the value function is estimated). Unfortunately quite few value function approximator allow deriving an uncertainty information about estimated values. Engel (2005) proposes such a model-free algorithm, but the actual use of value uncertainty is left as a perspective. In this paper, we show how some uncertainty information about estimated values can be derived from the Kalman Temporal Differences (KTD) framework of Geist et al. (2009a,b). We also introduce a form of active learning which uses this uncertainty information in order to speed up learning, as well as some adaptations of existing schemes designed to handle the exploration/exploitation dilemma. Each contribution is illustrated and experimented, the last one on a real-world dialogue management problem.

2. Background

2.1. Reinforcement Learning

This paper is placed in the framework of Markov decision process (MDP). An MDP is a tuple $\{S, A, P, R, \gamma\}$, where S is the state space, A the action space, $P : s, a \in S \times A \rightarrow p(.|s, a) \in \mathcal{P}(S)$ a family of transition probabilities, $R : S \times A \times S \rightarrow \mathbb{R}$ the bounded reward function, and γ the discount factor. A policy π associates to each state a probability over actions, $\pi : s \in S \rightarrow \pi(.|s) \in \mathcal{P}(A)$. The value function of a given policy is defined as $V^\pi(s) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, \pi]$ where r_i is the immediate reward observed at time step i, and the expectation is done over all possible trajectories starting in s given the system dynamics and the followed policy. The Q-function allows a supplementary degree of freedom for the first action and is defined as $Q^\pi(s, a) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, a_0 = a, \pi]$. RL aims at finding (through interactions) the policy π^* which maximises the value function for every state: $\pi^* = \text{argmax}_\pi(V^\pi)$. Two schemes among others can lead to the optimal policy. First, *policy iteration* involves learning the value function of a given policy and then improving the policy, the new one being greedy respectively to the learnt value function. It requires solving the *Bellman evaluation equation*, which is given here for the value and Q-functions: $V^\pi(s) = E_{s',a|\pi,s}[R(s, a, s') + \gamma V^\pi(s')]$ and $Q^\pi(s, a) = E_{s',a'|\pi,s,a}[R(s, a, s') + \gamma Q^\pi(s', a')]$. The second scheme, *value iteration*, aims directly at finding the optimal policy. It requires solving

the *Bellman optimality equation*: $Q^*(s,a) = E_{s'|s,a}[R(s,a,s') + \gamma \max_{b \in A} Q^*(s',b)]$. For large state and action spaces, exact solutions are tricky to obtain and value or $Q-$function approximation is required.

2.2. Kalman Temporal Differences - KTD

Originally, the Kalman (1960) filter paradigm is a statistical method aiming at online tracking the hidden state of a non-stationary dynamic system through indirect observations of this state. The idea behind KTD is to cast value function approximation into such a filtering paradigm: considering a function approximator based on a familly of parameterized functions, the parameters are then the hidden state to be tracked, the observation being the reward linked to the parameters through one of the classical Bellman equations. Thereby value function approximation can benefit from the advantages of Kalman filtering and particularly uncertainty management because of statistical modelling.

The following notations are adopted, given that the aim is the value function evaluation, the Q-function evaluation or the Q-function direct optimization:

$$t_i = \begin{cases} (s_i, s_{i+1}) \\ (s_i, a_i, s_{i+1}, a_{i+1}) \\ (s_i, a_i, s_{i+1}) \end{cases} \quad g_{t_i}(\theta_i) = \begin{cases} \hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) \\ \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \hat{Q}_{\theta_i}(s_{i+1}, a_{i+1}) \\ \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \max_b \hat{Q}_{\theta_i}(s_{i+1}, b) \end{cases} \tag{1}$$

where $\hat{V}_\theta$ (resp. $\hat{Q}_\theta$) is a parametric representation of the value (resp. Q-) function, θ being the parameter vector. A statistical point of view is adopted and the parameter vector is considered as a random variable. The problem at sight is stated in a so-called *state-space formulation*:

$$\begin{cases} \theta_i = \theta_{i-1} + v_i \\ r_i = g_{t_i}(\theta_i) + n_i \end{cases} \tag{2}$$

Using the vocabulary of Kalman filtering, the first equation is the evolution equation. It specifies that the searched parameter vector follows a random walk which expectation corresponds to the optimal estimation of the value function at time step i. The evolution noise v_i is centered, white, independent and of variance matrix P_{v_i}. The second equation is the observation equation, it links the observed transitions and rewards to the value (or Q-) function through one of the Bellman equations. The observation noise n_i is supposed centered, white, independent and of variance P_{n_i}.

KTD is a second order algorithm: it updates the mean parameter vector, but also the associated covariance matrix after each interaction. It breaks down into three steps. First, *predictions* of the parameters first and second order moments are obtained according to the evolution equation and using previous estimates. Then some *statistics of interest* are computed. The third step applies a *correction* to predicted moments of the parameters vector according to the so-called Kalman gain K_i (computed thanks to the statistics obtained in second step), the predicted reward $\hat{r}_{i|i-1}$ and the observed reward r_i (their difference being a form of temporal difference error).

Statistics of interest are generaly not analytically computable, except in the linear case. This does not hold for nonlinear parameterizations such as neural networks and for the Bellman optimality equation (because of the max operator). Nevertheless, a derivative-free

approximation scheme, the unscented transform (UT) of Julier and Uhlmann (2004), allows estimating first and second order moments of a nonlinearly mapped random vector. Let X be a random vector (typically the parameter vector) and $Y = f(X)$ its nonlinear mapping (typically the g_{t_i} function). Let n be the dimension of the random vector X. A set of $2n+1$ so-called sigma-points and associated weights are computed as follows:

$$\begin{cases} x^{(0)} = \bar{X} & j = 0 \\ x^{(j)} = \bar{X} + (\sqrt{(n+\kappa)P_X})_j & 1 \leq j \leq n \\ x^{(j)} = \bar{X} - (\sqrt{(n+\kappa)P_X})_{n-j} & n+1 \leq j \leq 2n \end{cases} \text{ and } \begin{cases} w_0 = \frac{\kappa}{n+\kappa} & j = 0 \\ w_j = \frac{1}{2(n+\kappa)} & 1 \leq j \leq 2n \end{cases} \quad (3)$$

where $\bar{X}$ is the mean of X, P_X is its variance matrix, κ is a scaling factor which controls the accuracy, and $(\sqrt{P_X})_j$ is the j^{th} column of the Cholesky decomposition of P_X. Then the image of each sigma-point through the mapping f is computed: $y^{(j)} = f(x^{(j)}), 0 \leq j \leq 2n$. The set of sigma-points and their images can then be used to compute the following approximations:$\bar{Y} \approx \bar{y} = \sum_{j=0}^{2n} w_j y^{(j)}$, $P_Y \approx \sum_{j=0}^{2n} w_j (y^{(j)} - \bar{y})(y^{(j)} - \bar{y})^T$ and $P_{XY} \approx \sum_{j=0}^{2n} w_j (x^{(j)} - \bar{X})(y^{(j)} - \bar{y})^T$.

Thanks to the UT, practical algorithms can be derived. At time-step i, a set of sigma-points is computed from predicted random parameters characterized by mean $\hat{\theta}_{i|i-1}$ and variance $P_{i|i-1}$. Predicted rewards are then computed as images of these sigma-points using one of the observation functions (1). Then sigma-points and their images are used to compute statistics of interest. This gives rise to a generic algorithm valid for any of the three Bellman equations and any parametric representation of V or Q summarized in Alg. 1, p being the number of parameters. More details as well as theoretical results (such as proofs of convergence) about KTD are provided by Geist and Pietquin (2010).

Algorithm 1: KTD

Initialization: priors $\hat{\theta}_{0|0}$ and $P_{0|0}$;

for $i \leftarrow 1, 2, \ldots$ **do**

Observe transition t_i and reward r_i;

Prediction Step;
$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;
$P_{i|i-1} = P_{i-1|i-1} + P_{v_i}$;

Sigma-points computation;
$\Theta_{i|i-1} = \{\hat{\theta}^{(j)}_{i|i-1}, 0 \leq j \leq 2p\}$ `/* from` $\hat{\theta}_{i|i-1}$ `and` $P_{i|i-1}$ `*/`
$\mathcal{W} = \{w_j, 0 \leq j \leq 2p\}$;
$\mathcal{R}_{i|i-1} = \{\hat{r}^{(j)}_{i|i-1} = g_{t_i}(\hat{\theta}^{(j)}_{i|i-1}), 0 \leq j \leq 2p\}$ `/* see Eq. (1)` `*/`

Compute statistics of interest;
$\hat{r}_{i|i-1} = \sum_{j=0}^{2p} w_j \hat{r}^{(j)}_{i|i-1}$;
$P_{\theta r_i} = \sum_{j=0}^{2p} w_j (\hat{\theta}^{(j)}_{i|i-1} - \hat{\theta}_{i|i-1})(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})$;
$P_{r_i} = \sum_{j=0}^{2p} w_j (\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})^2 + P_{n_i}$;

Correction step;
$K_i = P_{\theta r_i} P_{r_i}^{-1}$;
$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i (r_i - \hat{r}_{i|i-1})$;
$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

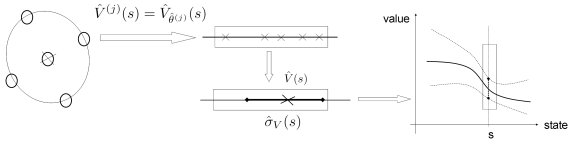


Figure 1: Uncertainty computation.

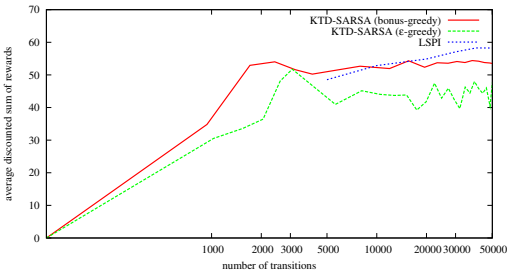


Figure 2: Dialog management results.

3. Computing Uncertainty over Values

The parameters being modeled as random variables, the parameterized value for any given state is a random variable. This model allows computing the mean and associated uncertainty. Let $\hat{V}_\theta$ be the approximated value function parameterized by the random vector θ of mean $\bar{\theta}$ and variance matrix P_θ. Let $\bar{V}_\theta(s)$ and $\hat{\sigma}^2_{V_\theta}(s)$ be the associated mean and variance for a given state s. To propagate the uncertainty from the parameters to the approximated value function a first step is to compute the sigma-points associated to the parameter vector, that is $\Theta = \{\theta^{(j)}, 0 \leq j \leq 2p\}$, as well as corresponding weights, from $\bar{\theta}$ and P_θ as described before. Then the images of these sigma-points are computed using the parameterized value function: $\mathcal{V}_\theta(s) = \{\hat{V}^{(j)}_\theta(s) = \hat{V}_{\theta^{(j)}}(s), \quad 0 \leq j \leq 2p\}$. Knowing these images and corresponding weights, the statistics of interest are computed: $\bar{V}_\theta(s) = \sum_{j=0}^{2p} w_j \hat{V}^{(j)}_\theta(s)$ and $\hat{\sigma}^2_{V_\theta}(s) = \sum_{j=0}^{2p} w_j (\hat{V}^{(j)}_\theta(s) - \bar{V}_\theta(s))^2$. This is illustrated on Fig. 1. Extension to Q-function is straightforward. So, as at each time-step uncertainty information can be computed in the KTD framework.

4. A Form of Active Learning

4.1. Principle

It is shown here how this available uncertainty information can be used in a form of active learning. The KTD algorithm derived from the Bellman optimality equation, that is Alg. 1 with third equation of Eq. (1), is named KTD-Q. It is an off-policy algorithm: it learns the optimal policy π^* while following a different behaviorial policy b. A natural question is: what behaviorial policy to choose so as to speed up learning? Let i be the current temporal index. The system is in a state s_i, and the agent has to choose an action a_i. The predictions $\hat{\theta}_{i|i-1}$ and $P_{i|i-1}$ are available and can be used to approximate the uncertainty of the Q-function parameterized by $\theta_{i|i-1}$ in the state s_i and for any action a. Let $\hat{\sigma}^2_{Q_{i|i-1}}(s_i, a)$ be the corresponding variance. The action a_i is chosen according to the following heuristic:

$$b(.|s_i) = \frac{\hat{\sigma}_{Q_{i|i-1}}(s_i, .)}{\sum_{a \in A} \hat{\sigma}_{Q_{i|i-1}}(s_i, a)} \tag{4}$$

This totally explorative policy favours uncertain actions. The corresponding algorithm which is called active KTD-Q (Alg. 1 with 3$^{\text{rd}}$ Eq. of (1) and policy (4)).

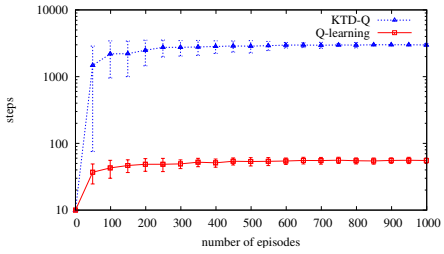


Figure 3: Optimal policy learning.

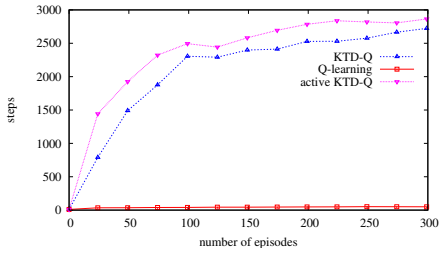


Figure 4: Random and active learning.

4.2. Experiment

The second experiment is the inverted pendulum benchmark. This task requires maintaining a pendulum of unknown length and mass at the upright position by applying forces to the cart it is attached to. It is fully described by Lagoudakis and Parr (2003) and we use the same parameterization (a mixture of Gaussian kernels). The goal is here to compare two value-iteration-like algorithms, namely KTD-Q and Q-learning, which aim at learning directly the optimal policy from suboptimal trajectories (off-policy learning). As far as we know, KTD-Q is the first second-order algorithm for Q-function approximation in a value iteration scheme, the difficulty being to handle the max operator (Yu and Bertsekas (2007) propose also such an algorithm, however for a restrictive class of MDP). That is why we compare it to a first-order algorithm. The active learning scheme is also experimented: it uses the uncertainty computed by KTD to speed up convergence.

For Q-learning, the learning rate is set to $\alpha_i = \alpha_0 \frac{n_0+1}{n_0+i}$ with $\alpha_0 = 0.5$ and $n_0 = 200$, according to Lagoudakis and Parr (2003). For KTD-Q, the parameters are set to $P_{0|0} = 10I$, $P_{n_i} = 1$ and $P_{v_i} = 0I$. For all algorithms the initial parameter vector is set to zero. Training samples are first collected online with a random behavior policy. The agent starts in a randomly perturbed state close to the equilibrium. Performance is measured as the average number of steps in an test episode (a maximum of 3000 steps is allowed). Results are averaged over 100 trials. Fig. 3 compares KTD-Q and Q-learning (the same random samples are used to train both algorithms). Fig. 4 adds active KTD-Q for which actions are sampled according to (4). Average length of episodes with totally random policy is 10, whereas it is 11 for policy (4). Consequently the increase in length can only slightly help to improve speed of convergence (at most 10%, much less than the real improvement which is about 100%, at least at the beginning).

According to Fig. 3, KTD-Q learns an optimal policy (that is balancing the pole for the maximum number of steps) asymptotically and near-optimal policies are learned after only a few tens of episodes (notice that these results are comparable to the LSPI algorithm). With the same number of learning episodes, Q-learning with the same linear parameterization fails to learn a policy which balances the pole for more than a few tens of time steps. Similar results for Q-learning are obtained by Lagoudakis and Parr (2003). According to Fig. 4, it is clear that sampling actions according to uncertainty speeds up convergence. It is almost doubled in the first 100 episodes. Notice that this active learning scheme could not have been used for Q-learning with value function approximation, as this algorithm cannot provide uncertainty information.

5. Exploration/Exploitation Dilemma

In this section, we present several approaches designed to handle the dilemma between exploration and exploitation (which can be linked to active learning). The first one is the well known ϵ-greedy policy, and it serves as a baseline. Other approaches are inspired from the literature and use the available uncertainty information (see Sec. 3 for its computation). The corresponding algorithms are a combination of KTD-SARSA (Alg. 1 with 2$^{\text{nd}}$ Eq. of (1)) with policies (5-8).

5.1. ϵ-greedy Policy

With an ϵ-greedy policy (Sutton and Barto, 1996), the agent chooses a greedy action respectively to the currently estimated Q-function with a probability $1-\epsilon$, and a random action with a probability ϵ (δ is the Kronecker symbol):

$$\pi(a_{i+1}|s_{i+1}) = (1-\epsilon)\delta(a_{i+1} = \underset{b\in A}{\operatorname{argmax}}\, \bar{Q}_{i|i-1}(s_{i+1},b)) + \epsilon\delta(a_{i+1} \neq \underset{b\in A}{\operatorname{argmax}}\, \bar{Q}_{i|i-1}(s_{i+1},b)) \quad (5)$$

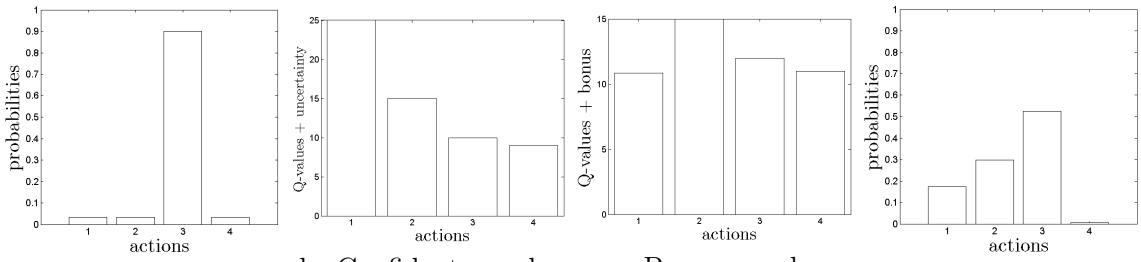
This policy is perhaps the most basic one, and it does not use any uncertainty information. An arbitrary Q-function for a given state and 4 different actions is illustrated on Fig. 6. For each action, it gives the estimated Q-value as well as the associated uncertainty (that is $\pm$ estimated standard deviation). For example, action 3 has the highest value and the lowest uncertainty, and action 1 the lowest value but the highest uncertainty. The probability distribution associated to the ϵ-greedy policy is illustrated on Fig. 5.a. The highest probability is associated to action 3, and other actions have the same (low) probability, despite their different estimated values and standard deviations.

5.2. Confident-greedy Policy

The second approach we propose consists in acting greedily according to the upper bound of an estimated confidence interval. The approach is not novel (Kaelbling, 1993), however some PAC (probably approximately correct) guarantees have been given recently by Strehl and Littman (2006) for a tabular representation (for which the confidence interval is proportional to the inverse of the square root of the number of visits to the considered state-action pair). In our case, we postulate that the confidence interval width is proportional to the estimated standard deviation (which is true if the parameters distribution is assumed to be Gaussian). Let α be a free positive parameter, we define the confident-greedy policy as:

$$\pi(a_{i+1}|s_{i+1}) = \delta\Big(a_{i+1} = \underset{b\in A}{\operatorname{argmax}} \left(\bar{Q}_{i|i-1}(s_{i+1},b) + \alpha\hat{\sigma}_{Q_{i|i-1}}(s_{i+1},b)\right)\Big) \quad (6)$$

The same arbitrary Q-values are considered (see Fig. 6), and the confident-greedy policy is illustrated on Fig. 5.b which represents the upper bound of the confidence interval. Action 1 is chosen because it has the highest score (despite the fact that it has the lowest estimated value). Notice that action 3, which is greedy respectively to the estimated Q-function, has only the third score.



a. ϵ-greedy. b. Confident-greedy. c. Bonus-greedy. d. Thompson.

Figure 5: Policies.

5.3. Bonus-greedy Policy

The third approach we propose is inspired from the method of Kolter and Ng (2009). The policy they use is greedy respectively to 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 (which can be interpreted as a variance, instead of the square-root of this quantity for interval estimation-based approaches which can be interpreted as a standard deviation). The bonus-greedy policy we propose uses the variance rather than the standard deviation, and is defined as (β_0 and β being two free parameters):

$$\pi(a_{i+1}|s_{i+1}) = \delta\Big(a_{i+1} = \underset{b\in A}{\text{argmax}} \left(\bar{Q}_{i|i-1}(s_{i+1},b) + \beta \frac{\hat{\sigma}^2_{Q_{i|i-1}}(s_{i+1},b)}{\beta_0 + \hat{\sigma}^2_{Q_{i|i-1}}(s_{i+1},b)}\right)\Big) \tag{7}$$

The bonus-greedy policy is illustrated on Fig. 5.c, still using the arbitrary Q-values and associated standard deviations of Fig. 6. Action 2 has the highest score, it is thus chosen. Notice that the three other actions have approximately the same score, despite the fact that they have quite different Q-values.

5.4. Thompson Policy

Recall that the KTD algorithm maintains the parameters mean vector and variance matrix. Assuming that the parameters distribution is Gaussian, we propose to sample a set of parameters from this distribution, and then to act greedily according to the resulting sampled Q-function. This type of scheme was first proposed by Thompson (1933) for a bandit problem, and it has been recently introduced into the reinforcement learning community in the tabular case (Dearden et al., 1998; Strens, 2000). Let the Thompson policy be:

$$\pi(a_{i+1}|s_{i+1}) = \underset{b\in A}{\text{argmax}}\, \hat{Q}_\xi(s_{i+1},b) \text{ with } \xi \sim \mathcal{N}(\hat{\theta}_{i|i-1}, P_{i|i-1}) \tag{8}$$

We illustrate the Thompson policy on Fig. 5.d by showing the distribution of the greedy action (recall that parameters are random, and thus the greedy action too). The highest probability is associated to action 3. However, notice that a highest probability is associated to action 1 than to action 4: the first one has a lower estimated Q-value, but it is less certain.

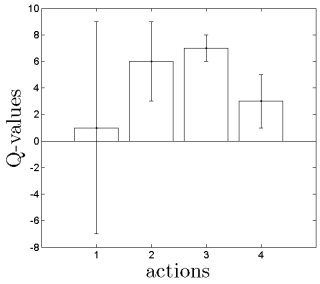


Figure 6: Q-values and associated uncertainty.

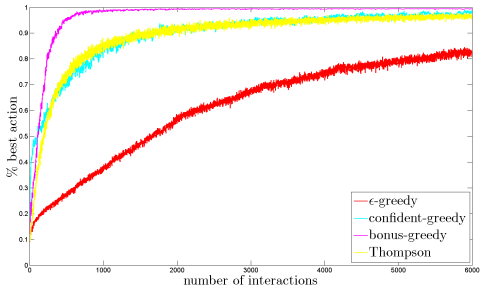


Figure 7: Bandit results.

5.5. Experiment

The bandit problem is an MDP with one state and N actions. Each action a implies a reward of 1 with probability p_a, and a reward of 0 with probability $1 - p_a$. For an action a^* (randomly chosen at the beginning of each experiment), the probability is set to $p_{a^*} = 0.6$. For all other actions, the associated probability is uniformly and randomly sampled between 0 and 0.5: $p_a \sim \mathcal{U}_{[0,0.5]}, \forall a \neq a^*$. Presented results are averaged over 1000 experiments. The performance of a method is measured as the percentage of time the optimal action has been chosen, given the number of interactions between the agent and the bandit. A tabular representation is adopted for KTD-SARSA, and the following parameters are used[1]: $N = 10$, $P_{0|0} = 0.1I$, $\theta_{0|0} = I$, $P_{n_i} = 1$, $\epsilon = 0.1$, $\alpha = 0.3$, $\beta_0 = 1$ and $\beta = 10$. As the considered bandit has $N = 10$ arms, a random policy has a performance of 0.1. Notice also that a purely greedy policy would choose systematically the first action for which the agent has observed a reward.

Results presented in Fig. 7 compare the four schemes. The ϵ-greedy policy serves as a baseline, and all proposed schemes using the available uncertainty performs better. Thompson policy and confident-greedy policy perform approximately equally well, and the best results are obtained by the bonus-greedy policy. Of course, these quite preliminary results do not allow to conclude about guarantees of convergence of the proposed schemes. However, they tend to show that the computed uncertainty information is meaningful and that it can provide useful for the dilemma between exploration and exploitation.

6. Dialogue management application

In this section is proposed an application to a real world problem: spoken dialogue management. A spoken dialog system (SDS) generally aims at providing information to a user through natural language-based interactions. An SDS has roughly three modules: a speech understanding component (speech recognizer and semantic parser), a dialogue manager and a speech generation component (natural language generator and speech synthesis). Dialogue management is a sequential decision making problem where a dialogue manager has to select

1. For an empirical study of the sensitivity of performance of the proposed policies as a function of parameter setting, see Geist (2009).

which information should be asked or provided to the user when in a given situation. It can thus be cast into the MDP framework (Levin et al., 2000; Singh et al., 1999; Pietquin and Dutoit, 2006). The set of *actions* a dialog manager can select is defined by so called *dialog acts*. There can be different 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 user responses. It summarizes the information exchanged between the user and the system until the considered state is reached. A dialogue management strategy 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.

The considered system is a form-filling spoken dialog system. It is oriented toward tourism information, similarly to the one described by Lemon et al. (2006). Its goal is to provide information about restaurants based on specific user preferences. There are three slots in this dialog problem, namely the location of the restaurant, the cuisine type of the restaurant and its price-range. Given past interactions with the user, the agent asks a question so as to propose the best choice according to the user preferences. The goal is to provide the correct information to the user with as few interactions as possible. 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 automatic 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 the dialog is closed. 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 $\gamma = 0.95$. Even if the ultimate goal is to implement RL on a real dialog management problem, in this experiment a user simulation technique was used to generate data (Pietquin and Dutoit, 2006). The user simulator was plugged to the DIPPER dialogue management system (Lemon et al., 2006) to generate dialogue samples. The Q-function is represented using one 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.

KTD-SARSA with ϵ-greedy and bonus-greedy policies are compared on Fig.2 (results are averaged over 8 independent trials, and each point is averaged over 100 past episodes: a stable curve means a low standard deviation). LSPI, a batch and off-policy approximate policy iteration algorithm (Lagoudakis and Parr, 2003), serves as a baseline. It was trained in an off-policy and batch manner using random trajectories, and this algorithm provide competitive results among the state of the art (Li et al., 2009a; Chandramohan et al., 2010). Both algorithms provide good results (a positive cumulative reward, which means that the user is generally satisfied after few interactions). However, one can observe that the bonus-greedy scheme provides faster convergence as well as better and more stable policies than the uninformed ϵ-greedy policy. Moreover, results for the informed KTD-SARSA are very

close to LSPI after few learning episodes. Therefore, KTD-SARSA is sample efficient (it provides good policies while the insufficient number of transitions prevents from using LSPI because of numerical stability problems), and the provided uncertainty information is useful on this dialogue-management task.

7. Conclusion

In this paper, we have shown how an uncertainty information about estimated values can be derived from KTD. We have also introduced an active learning scheme aiming at improving speed of convergence by sampling actions according to their relative uncertainty, as well as some adaptations of existing schemes for exploration/exploitation. Three experiments have been proposed. The first one shown that KTD-Q, a second-order value-iteration-like algorithm, is sample efficient. The improvement gained by using the proposed active learning scheme was also demonstrated. The proposed schemes for exploration/exploitation were also successfully experimented on a bandit problem and the bonus-greedy policy on real-world problem. This is a first step toward combining the dilemma between exploration and exploitation with value function approximation.

The next step is to adapt more existing approaches dealing with the exploration/exploitation dilemma designed for tabular representation of the value function to the KTD framework, and to provide some theoretical guarantees for the proposed approaches. This paper focused on model-free reinforcement learning, and we plan to compare our approach to model-based RL approaches.

Acknowledgments

The authors thank the European Community (FP7/2007-2013, grant agreement 216594, CLASSiC project : www.classic-project.org) and the Région Lorraine for financial support.

References

S. Chandramohan, M. Geist, and O. Pietquin. Sparse Approximate Dynamic Programming for Dialog Management. In *Proceedings of the 11th SIGDial Conference on Discourse and Dialogue*, pages 107–115, Tokyo (Japan), September 2010. ACL.

R. Dearden, N. Friedman, and S. J. Russell. Bayesian Q-Learning. In *AAAI/IAAI*, pages 761–768, 1998.

Y. Engel. *Algorithms and Representations for Reinforcement Learning*. PhD thesis, Hebrew University, April 2005.

M. Geist. *Optimisation des chaînes de production dans l'industrie sidérurgique : une approche statistique de l'apprentissage par renforcement*. Phd thesis in mathematics, Université Paul Verlaine de Metz (en collaboration avec Supélec, ArcelorMittal et l'INRIA), Novembre 2009.

M. Geist and O. Pietquin. Kalman Temporal Differences. *Journal of Artificial Intelligence Research (JAIR)*, 2010.

M. Geist, O. Pietquin, and G. Fricout. Kalman Temporal Differences: the deterministic case. In *IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2009)*, Nashville, TN, USA, April 2009a.

M. Geist, O. Pietquin, and G. Fricout. Tracking in reinforcement learning. In *International Conference on Neural Information Processing (ICONIP 2009)*, Bangkok (Thailand), December 2009b. Springer.

N. Jong and P. Stone. Model-Based Exploration in Continuous State Spaces. In *Symposium on Abstraction, Reformulation, and Approximation*, July 2007.

S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92 (3):401–422, 2004.

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

S. Kakade, M. J. Kearns, and J. Langford. Exploration in Metric State Spaces. In *International Conference on Machine Learning (ICML 03)*, pages 306–312, 2003.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

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, 2009. ACM.

M. G. Lagoudakis and R. Parr. Least-Squares Policy Iteration. *Journal of Machine Learning Research*, 4: 1107–1149, 2003.

S. Larsson and D. R. Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 2000.

O. Lemon, K. Georgila, J. Henderson, and M. Stuttle. An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system. In *Meeting of the European chapter of the Associaton for Computational Linguistics (EACL'06)*, Morristown, NJ, USA, 2006.

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*, 8(1):11–23, 2000.

L. Li, S. Balakrishnan, and J. Williams. Reinforcement Learning for Dialog Management using Least-Squares Policy Iteration and Fast Feature Selection. In *Proceedings of the International Conference on Speech Communication and Technologies (InterSpeech'09)*, Brighton (UK), 2009a.

L. Li, M. Littman, and C. Mansley. Online exploration in least-squares policy iteration. In *Conference for research in autonomous agents and multi-agent systems (AAMAS-09)*, Budapest, Hungary, 2009b.

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, March 2006.

Y. Sakaguchi and M. Takano. Reliability of internal prediction/estimation and its application: I. adaptive action selection reflecting reliability of value function. *Neural Networks*, 17(7):935–952, 2004.

S. Singh, M. Kearns, D. Litman, and M. Walker. Reinforcement learning for spoken dialogue systems. In *Conference on Neural Information Processing Society (NIPS'99), Denver, USA*. Springer, 1999.

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.

M. Strens. A Bayesian Framework for Reinforcement Learning. In *International Conference on Machine Learning*, pages 943–950. Morgan Kaufmann, San Francisco, CA, 2000.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1996.

W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of two samples. *Biometrika*, (25):285–294, 1933.

H. Yu and D. P. Bertsekas. Q-Learning Algorithms for Optimal Stopping Based on Least Squares. In *European Control Conference*, Kos, Greece, 2007.

Optimising a Handcrafted Dialogue System Design

Romain Laroche[1]*, Ghislain Putois*[1] *and Philippe Bretier*[1]

[1]Orange Labs, Issy-les-Moulineaux, France

romain.laroche@orange-ftgroup.com

Abstract

In the Spoken Dialogue System literature, all studies consider the dialogue move as the unquestionable unit for reinforcement learning. Rather than learning at the dialogue move level, we apply the learning at the design level for three reasons : 1/ to alleviate the high-skill prerequisite for developers, 2/ to reduce the learning complexity by taking into account just the relevant subset of the context and 3/ to have interpretable learning results that carry a reusable usage feedback. Unfortunately, tackling the problem at the design level breaks the Markovian assumptions that are required in most Reinforcement Learning techniques. Consequently, we decided to use a recent non-Markovian algorithm called Compliance Based Reinforcement Learning. This paper presents the first experimentation on online optimisation in dialogue systems. It reveals a fast and significant improvement of the system performance with by average one system misunderstanding less per dialogue.

Index Terms : Spoken Dialogue Systems, Reinforcement Learning, Online Learning, Hybrid System

1. Introduction

Spoken Dialogue Systems (SDS) with learning capabilities have been deeply investigated these last ten years [1]. The primary goal pursued by these studies is to reduce the development cost by automating the SDS design. They use Markov Decision Process (MDP) to learn the best dialogue move according to the current dialogue state. This approach has led to promising results but, as some researchers [2] acknowledge, some obstacles are difficult to overcome (Reinforcement Learning skill prerequisite, simulated users, ...). The high dimensionality of the dialogue states and actions is the main reason for all these troubles. As a consequence, reducing this space complexity represents a big part of the scientific effort with for instance hierarchy of states [3] and summary of states [4]. In fact, these works endeavour to improve the system performance by inserting expert knowledge.

Other studies [5, 6] propose to mix a handcrafted SDS with an MDP based SDS. The goal is completely different : to optimise the dialogue capabilities of a conventional SDS. In substance, the idea consists in handcrafting almost totally the SDS so that it provides a small collection of options among which the MDP based SDS picks the action to generate to the user. We completely agree with this objective. We endeavour to improve these previous works by using a novel framework that learns at the handcrafting design level (which question, which words, which prosody, ...), instead of the dialogue move level. This enables to drastically reduce the dimensionality of the learning and to consequently speed up the convergence [7].

Section 2 gives a detailed description of the problem and recalls the Compliance Based Reinforcement Learning algorithm [7] that we use for design optimisation. Then, section 3 describes the experimented system and the goal of the experimentation. Next, section 4 presents the experimentation results. Finally, section 5 concludes the paper with the next steps.

2. Problem Constraints and Theoretical Resolution

This section first recalls how conventional SDS are designed by developers. Then, it explains why the Markovian assumption cannot be maintained in this design environment. Thus, the Module-Variable-Decision Process is introduced. Finally, the section ends with the presentation of the Compliance-Based Reinforcement Learning : a reinforcement learning algorithm that does not use the Markovian assumption.

2.1. The conventional SDS design environment

Industry follows the VUI-completeness principle [8] : *"the behaviour of an application needs to be completely specified with respect to every possible situation that may arise during the interaction. No unpredictable user input should ever lead to unforeseeable behaviour"*. The SDS developers consider reliable the technologies, tools, and methodologies that help them to reach the VUI-completeness and to control it.

The graphical abstraction used for SDS design conforms to the general graph representation of finite state automata, with the difference that global and local variables (or context) enable to factorise several system states in a single node. A system state is therefore described by the automata node and the global context aggregating the global context with all the local contexts. Transitions relate to user inputs or to internal application events such as conditions based on internal information from the current dialogue state, from the back-end, or from the dialogue history. In that sense, dialogue design in the industry generally covers more than strict dialogue management, since its specification may indicate the type of spoken utterance expected from the user at each stage of the dialogue, up to the precise speech recognition model and parameter values to use, and the generation of the system utterance, from natural language generation to speech synthesis or audio recordings.

2.2. Problem Constraints

Most dialogue moves can be split up into several internal decisions that are represented by as many automata nodes : type of feedback, insertion of contextual help or presentation of additional information and choices of questions. These internal decisions are conventionally specified with handcrafted rules designed from a very localised view of the system state. For instance, the system's decision to welcome the user with "good morning" or "good afternoon" will depend solely on the current time at the place from where the user calls. Another more complex example : the decision to insert one help message depends

on the expected expertise of the user. It is easy to understand that this expected expertise depends on the service users' average expertise and on the number of errors that occurred during the current dialogue. However, it is very complex for the developer to estimate the users average expertise or to design a function between those context elements and the decision to make. This is an example where the decision has to be based on statistics. The developer designs the decision alternatives and the relevant information to take into account for making this decision. Learning at a decision level complies with the way developers are currently designing their application.

Learning at a decisional level is also motivated by the learning improvement demonstrated on simulated examples in [7]. Indeed, it enables to use for each decision only the precise relevant context, instead of taking into account the whole dialogue context. This leads to a space reduction and therefore to a faster convergence.

In addition to providing a tool for experimenting several alternatives and optimising the system's behaviour, this approach offers reporting capabilities. In fact, it delivers a usage feedback inside the very dialogue automata the developer designed in the first place. Contrary to MDP-based methods that flourish in the literature, our method enables the explanation and thus the understanding of what the system learnt. Then, this inferred knowledge can be reused in a another context or even another dialogue application. This ability to generate expert knowledge acquired with online learning is new in SDS literature.

Unfortunately, in spite of these advantages, our approach has a strong technical shortcoming : it breaks the Markovian assumptions. Indeed, at a time t, the decision process state s_t is described by the automata node m_t and the local context v_t taken into account for making the decision. The next decision process state s_{t+1} cannot be forecast because its local context v_{t+1} is generally not included into s_t. As the decision process is non-Markovian, in a given decision process state, we cannot assume to be able to anticipate the next reached decision process state. As a consequence, we cannot use any bootstrapping method [9], *i.e.* any method that updates expectation estimates on the basis of the estimates of the following decision process states. The fastest reinforcement learning algorithms use bootstrapping, such as Dynamic Programming or Temporal Difference Learning.

2.3. Module-Variable Decision Process

This subsection recalls a recent framework [7] for learning at a decision level. Contrarily to MDP that consider the system state as a whole, in this framework, the system global state is not represented in the decision process. The decision process states are the locally relevant subset of the information included in the system state. Further in this paper, a *state* refers to the decision process state and the global system state concept is abandoned.

A *module* is the terminology we use for an automata node. For practical purpose, it is a processing unit that can execute an internal *action* according to its local *context*. This leads to the definition of the Module-Variable Decision Process (MVDP) framework (M, V_M, A_M) where :

- M is the set of modules.
- $\forall m \in M$, V_m is the local context used in module m.
- $\forall m \in M$, A_m is the set of possible actions for m.

In our previous example, we called s_t the decision process state at time t. This state is a tuple made of the module m_t accessed at time t and its local context v_t. Each module has a *policy*. The policy governs the choices that are made when reaching the corresponding automata node given a local context. A policy is a function from the context space into the action space $\pi_m : V_m \mapsto A_m$. In order to build its policy, the module may generate a *state-action value function* $Q_m : V_m \times A_m \mapsto \mathbb{R}$ which intends to predict the dialogue-term reward given the local context and the chosen action. The exploitation policy aims to maximise the dialogue-term reward expectations after a given decision d :

$$r_d = \sum_k \gamma^{t_k - t_d} R_k \tag{1}$$

Where $\gamma \in [0, 1]$ is the discount factor, used in order to encourage the shortest path to a dialogue success, t_d is the time when decision d has been made and $t_k > t_d$ is the time when reward R_k has been received.

2.4. Compliance Based Reinforcement Learning

A *decision* d has the following features : the module m where d is made, the local context v reduced to the relevant information concerning d, chosen action a and timestamp t.

$$d = (m, v, a, t) \in M \times V_m \times A_m \times \mathbb{R} \tag{2}$$

From the reinforcement learning algorithm point of view, an *episode* is the chain of decisions and rewards generated during a dialogue. The CBRL idea consists in avoiding to learn that an upfront decision is bad because the episode that tried it made further bad decisions. The basic idea of the algorithm is to avoid that an episode (d_a, d_b) where d_b is bad (and thus engendered a low reward) leads the system to learn that d_a is bad too based on this episode poor performance. In order to prevent this, the algorithm rates to what extent each episode is reliable for learning. Thus, the CBRL may consider that an episode should not be taken into account for evaluating a decision d_a because the further decisions are considered not good enough. This rating $c_\pi(e) \in \mathbb{R}^-$ is called the *compliance* of an episode e with the policy π. It represents the deviation of the episode e decisions from the policy π and it is computed as follows :

$$c_\pi(e) = \sum_{k=1}^{|e|} \gamma^{t_k - t_0} \left(Q_{m_k}(v_k, a_k) - \sup_{a \in A_{m_k}} Q_{m_k}(v_k, a) \right) \tag{3}$$

This compliance measures how well a decision has been evaluated by computing how compliant the further decisions of the episode are according to the current policy. Once the decision corpus $C = \{m_k, v_k, a_k, r_k, c_k = c_\pi(e_k)\}$ is generated (e_k is the episode after making decision d_k), the Monte Carlo method [9] is adapted to accept weighted average on the returns. Therefore, Q expectation is computed as follows :

$$C_{m,v,a} = \{r_k, c_k\} \text{ with } \{m, v, a, r_k, c_k\} \in C \tag{4}$$

$$Q_m(v, a) = \frac{\sum_{\{r_k, c_k\} \in C_{m,v,a}} r_k e^{\tau c_k}}{\sum_{\{r_k, c_k\} \in C_{m,v,a}} e^{\tau c_k}} \tag{5}$$

Where τ is a parameter expressing the impact level of the compliance on the weights.

3. Experimentation Settings

Our experimentation is a proof of concept on a small set of users before implementing these learning capabilities on a commercial system receiving dozens of thousands calls per month. In order to observe an improvement of the system after less than 200 calls, we had to reduce the alternative sets to four design points and also to limit the local context of each point to the empty set. With these limitations, an MDP system would probably have performed as well. The goal of this experimentation was not to show the performance superiority of the CBRL approach, which has been proven in a previous paper [7]. The goal was to prove that it was possible to improve dramatically a system's performance by optimising it through a simple and reduced set of alternatives.

We insist that the goal was not to learn a reusable policy as with batch learning which are trained with simulated users (even if it's technically possible). Indeed, our algorithm is designed for online learning, *i.e.* learning during runtime and our experimentation shows how a commercial system would improve during its lifetime, on the basis of its interactions with real customers.

The experimented system helps the user to install her DSL-box (called Livebox). The system and the evaluation forms were in French only, but for an easier understanding, all the information is here translated into English.

3.1. Implementation

As carrying out the Livebox installation by phone is a bit clunky, the system first tries to find another means to help the user with her Livebox installation. As a result, during the first part of the service, the system asks whether the user wants a technician to install her Livebox (if she does, the user is directed to the appointment scheduling service), whether the user is at home (if not, the user is asked to call again when she is) and whether the user has access to the internet (if she does, the system provides her with a URL where she can find the Livebox installation process). Once the system has checked that it could not bypass the installation process, the hardware installation process is described to the user. The installation process involves the plugging of the power cable, the DSL cable, the Ethernet cable and the DSL filters.

3.2. Proposed Alternatives

Instead of designing a completely deterministic service, we experimented several alternatives at four locations (or modules in the MVDP framework, see section 2.3) in the design. As we knew that the dialogue corpus would be limited to 160 units, we decided not to condition on context space V_m. The set of alternatives A_m are the following ones :

Orange Labs state-of-the-art unit selection speech synthesizer[1] was used with different acoustic inventories of the same professional female speaker to generate acoustic variants of the greeting message : neutral, calm and expressive.

Two orders for the bypassing questions were experimented : 1) home/internet/technician and 2) technician/home/internet.

Two natural language generation variants of the "are you at home" message were tested : a directive variant ("I would like to know if you are at home now.") and an interrogative variant ("Are you now at home ?"). Three speaking style TTS variants were tested for the interrogative variant : neutral, expressive and calm.

[1] demonstrator available at http ://tts.elibel.tm.fr

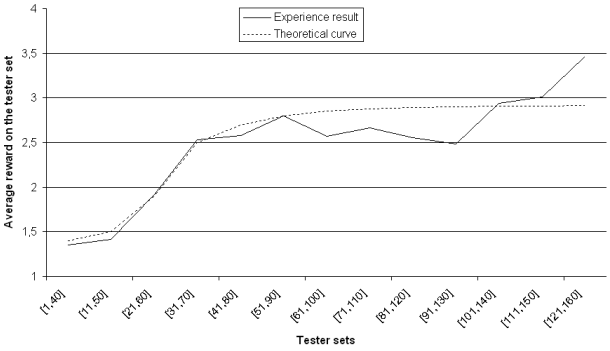


FIG. 1 – Overall performance evolution

During the installation process, there are a lot of information to present to the user : how they should plug the power cable, the DSL cable, the Ethernet cable, the ToIP adapter (which is not used in the installation process) and the DSL filters. There is no straightforward relevant order for these procedures. For instance, we have considered that the user might be more comfortable if the system first ask her to identify each element, before undertaking the plugging. Eventually, five different strategies have been tested.

4. Evaluation and Analysis

The system evaluation was completed using 40 subjects each performing four scenarios, corresponding to each three bypassing questions plus the full installation procedure.

4.1. System Auto-evaluation

4.1.1. Settings

The System auto-evaluation is the metrics used as rewards for the CBRL. In this experiment, they were task-based as follows :

- -1 for each ASR/SLU reject or time-out
- -10 for a hang-up or after an unsuccessful Livebox installation
- +5 after providing the user with an alternative way to install her Livebox.
- +10 after a successful Livebox installation

4.1.2. Overall Performance Evolution

The experimental results in figure 1 show that the learning algorithm choices triggered an overall dialogue performance improvement through time in a window of 40 dialogues.

The flat performance at the first stage can be explained by the exploration policy that let the first testers experience an almost fully exploration strategy, to be able to rate the fully exploratory policy. Then, around abscissa points [11,50] and [31,70], there follows a strong performance improvement probably due to what the learning algorithm learnt. The small decreasing of the plot around the [81,120] abscissa is probably due to testers that were performing under average. Similarly, the very high average rewards obtained at the end of the experimentation should be put into perspective with the fact that no learning has been done at this time, and this improvement can only be explained by users performing over the average. The dotted curve shows how we can extrapolate the system's performance

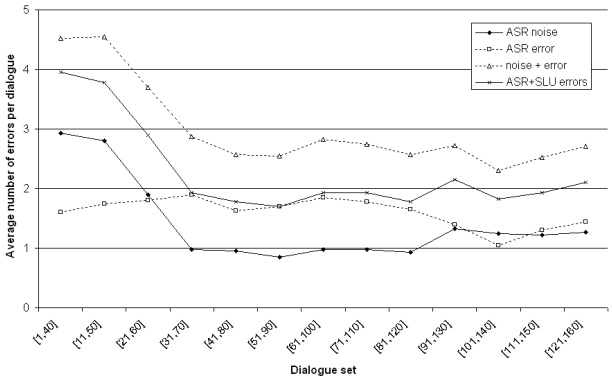


FIG. 2 – Evolution of the error rate based KPIs

expectations over time. This results shows that very similar systems can vary a lot in performance. It also shows that our algorithm significantly improved a handcrafted system in a very short time.

4.2. Objective Evaluation

In order to prove that reward's optimisation led to a real objective improvement of the system, we made two sets of objective Key Performance Indicators (KPI) : the duration based KPI such as call duration in seconds and number of dialogue turns, and the error rate based KPI such as ASR and SLU error rate. The duration based KPI were obtained automatically, while the error rate KPI were based on manual dialogue annotations.

Experimentation showed that the duration based KPI are not significantly connected with the dialogue performance as defined with the system rewards.

Concerning the error-based KPI, we considered ASR noise (ASR errors caused by the surrounding noise, the Livebox manipulation triggered a lot of them), ASR errors, ASR noise+error (the sum of the two previous KPI) and ASR+SLU errors (errors after the ASR+SLU chain).

Figure 2 shows that the main KPI to be affected by the system's learning is the ASR noise. That is how the choice of the best alternative expresses its improvement. To the contrary, the ASR errors do not look much affected by the learning. The error rates for ASR and ASR+SLU are directly influenced by the ASR noise curve and eventually the average number of errors per dialogue is divided by 2, which constitutes a strong result.

5. Conclusion

This paper presented the first experimentation in a real Spoken Dialogue System optimising online. It recalled the MVDP framework and the CBRL algorithm that were used for the application design. An evaluation on the Livebox installation applicatin was made : auto-evaluation of the system and objective evaluation. The auto-evaluation results showed that the break of the Markovian assumption did not endanger the convergence of the learning algorithm. The objective evaluation confirmed that the reward optimisation led to a dialogue error cut, and that the system really performed better at the end of the experimentation than at the beginning.

This paper proved the validity of our approach for optimisation in SDS with a significant error reduction along the learning. In addition, the experimentation showed other less quantifiable advantages of our approach, when compared to dialogue move optimisation. First, it is easier to implement. The applicative implementation does not require any technical skills. The developer just has to define the alternatives, the local context and the rewards, as opposed to dialogue turn based learning that suppose a summary/hierarchy of the dialogue state set. Second, although dialogue move learning only provides a general optimisation that cannot be interpreted or explained, our approach provides a valuable usage return for the application designers. In addition to that, a call flow monitoring tool has been integrated to the design studio, so that the developers or project managers can overview how each alternative performed and how they correlate to key performance indicators.

6. Acknowledgment

This research has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 216594 (CLASSIC project : www.classic-project.org).

7. References

[1] O. Lemon and O. Pietquin, "Machine learning for spoken dialogue systems," in *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)*, August 2007, pp. 2685–2688.

[2] T. Paek and R. Pieraccini, "Automating spoken dialogue management design using machine learning : An industry perspective," *Speech Communication*, vol. 50, pp. 716–729, 2008.

[3] H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira, "Reinforcement learning of dialogue strategies with hierarchical abstract machines," in *Proceedings of IEEE/ACL Workshop on Spoken Language Technology (SLT)*, December 2006.

[4] J. D. Williams and S. Young, "Scaling up POMDPs for dialog management : The summary POMDP method," in *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*, 2005, pp. 177–182.

[5] S. Singh, D. Litman, M. Kearns, and M. Walker, "Optimizing Dialogue Management with Reinforcement Learning : Experiments with the NJFun System." *Journal of Artificial Intelligence Research*, vol. 16, pp. 105–133, 2002. [Online]. Available : http ://www.jair.org/media/859/live-859-1983-jair.pdf

[6] J. D. Williams, "The best of both worlds : Unifying conventional dialog systems and POMDPs," in *International Conference on Speech and Language Processing*, 2008.

[7] R. Laroche, G. Putois, P. Bretier, and B. Bouchon-Meunier, "Hybridisation of expertise and reinforcement learning in dialogue systems," in *Proceedings of Interspeech. Special Session : Machine Learning for Adaptivity in Spoken Dialogue*, Brighton (United Knigdom), September 2009.

[8] R. Pieraccini and J. Huerta, "Where do we go from here ? research and commercial spoken dialog systems," in *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, L. Dybkjaer and W. Minker, Eds., 2005, pp. 1–10.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning : An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998. [Online]. Available : http ://www.amazon.ca/exec/obidos/redirect ?tag=citeulike09-20&amp ;path=ASIN/0262193981

Online Reinforcement Learning for Spoken Dialogue Systems: The Story of a Commercial Deployment Success

Ghislain Putois
Orange Labs
Lannion, France

Romain Laroche
Orange Labs
Issy-les-Moulineaux, France

Philippe Bretier
Orange Labs
Lannion, France

firstname.surname@orange-ftgroup.com

Abstract

Building an industrial spoken dialogue system (SDS) requires several iterations of design, deployment, test, and evaluation phases. Most industrial SDS developers use a graphical tool to design dialogue strategies. They are critical to get good system performances, but their evaluation is not part of the design phase.

We propose integrating dialogue logs into the design tool so that developers can jointly monitor call flows and their associated Key Performance Indicators (KPI). It drastically shortens the complete development cycle, and offers a new design experience.

Orange Dialogue Design Studio (ODDS), our design tool, allows developers to design several alternatives and compare their relative performances. It helps the SDS developers to understand and analyse the user behaviour, with the assistance of a reinforcement learning algorithm. The SDS developers can thus confront the different KPI and control the further SDS choices by updating the call flow alternatives.

Index Terms : Dialogue Design, Online Learning, Spoken Dialogue Systems, Monitoring Tools

1 Introduction

Recent research in spoken dialogue systems (SDS) has called for a "synergistic convergence" between research and industry (Pieraccini and Huerta, 2005). This call for convergence concerns architectures, abstractions and methods from both communities. Under this motivation, several research orientations have been proposed. This paper discusses three of them : dialogue design, dialogue management, and dialogue evaluation. Dialogue design and dialogue management reflect in this paper the respective paths that industry and research have followed for building their SDS. Dialogue evaluation is a concern for both communities, but remains hard to put into operational perspectives.

The second Section presents the context and related research. The third Section is devoted to the presentation of the tools : the historical design tool, its adaptation to provide monitoring functionalities and the insertion of design alternatives. It is eventually concluded with an attempt to reassessing the dialogue evaluation. The fourth Section describes the learning integration to the tool, the constraints we impose to the learning technique and the synergy between the tools and the embedded learning capabilities. Finally, the last Section concludes the paper.

2 Context

The spoken dialogue industry is structured around the architecture of the well known industrial standard VoiceXML [1]. The underlying dialogue model of VoiceXML is a mapping of the simplistic turn-based linguistic model on the browser-server based Web architecture (McTear, 2004). The browser controls the speech engines (recognition and text-to-speech) integrated into the voice platform according to the VoiceXML document served by an application server. A VoiceXML document contains a set of prompts to play and the list of the possible interactions the user is supposed to have at each point of the dialogue. The SDS developers [2], reusing Web standards and technologies (e.g. J2EE, JSP, XML...), are used to designing directed dialogues modelled by finite state automata. Such controlled and

1. http ://www.w3c.org/TR/voicexml20/
2. In this paper, the term "SDS developers" denotes without any distinction VUI designers, application developers, and any industry engineers acting in SDS building.

deterministic development process allows the spoken dialogue industry to reach a balance between usability and cost (Paek, 2007). This paper argues that tools are facilitators that improve both the usability vs. cost trade-off and the reliability of new technologies.

Spoken dialogue research has developed various models and abstractions for dialogue management : rational agency (Sadek et al., 1997), Information State Update (Bos et al., 2003), functional models (Pieraccini et al., 2001), planning problem solving (Ferguson and Allen, 1998). Only a very small number of these concepts have been transferred to industry. Since the late 90's, the research has tackled the ambitious problem of automating the dialogue design (Lemon and Pietquin, 2007), aiming at both reducing the development cost and optimising the dialogue efficiency and robustness. Recently, criticisms (Paek and Pieraccini, 2008) have been formulated and novel approaches (Williams, 2008) have been proposed, both aiming at bridging the gap between research –focused on Markov-Decision-Process (Bellman, 1957) based dialogue management– and industry –focused on dialogue design process, model, and tools. This paper contributes to extend this effort. It addresses all these convergence questions together as a way for research and industry to reach a technological breakthrough.

Regarding the dialogue evaluation topic, Paek (Paek, 2007) has pointed out that while research has exerted attention about "how best to evaluate a dialogue system ?", the industry has focused on "how best to design dialogue systems ?". This paper unifies those two approaches by merging system and design evaluation in a single graphical tool. To our knowledge, ODDS is the only industrial tool which handles the complete system lifecycle, from design to evaluation.

The tools and methods presented below have been tested and validated during the design and implementation of a large real-world commercial system : the 1013+ service is the Spoken Dialogue System for landline troubleshooting for France. It receives millions of calls a year and schedules around $8,000$ appointments a week. When the user calls the system, she is presented with an open question asking her for the reason of her call. If her landline is out of service, the Spoken Dialogue System then performs some automated tests on the line, and if the problem is confirmed, try and schedule an appointment with the user for a manual intervention. If the system and the user cannot agree on an appointment slot, the call is transferred to a human operator.

3 The tools

Industry follows the VUI-completeness principle (Pieraccini and Huerta, 2005) : *"the behaviour of an application needs to be completely specified with respect to every possible situation that may arise during the interaction. No unpredictable user input should ever lead to unforeseeable behaviour"*. The SDS developers consider reliable the technologies, tools, and methodologies that help them to reach the VUI-completeness and to control it.

3.1 The Dialogue Design Tool

The graphical abstraction proposed by our dialogue design tool conforms to the general graph representation of finite state automata, with the difference that global and local variables enable to factorise several dialogue states in a single node. Transitions relate to user inputs or to internal application events such as conditions based on internal information from the current dialogue state, from the back-end, or from the dialogue history. In that sense, dialogue design in the industry generally covers more than strict dialogue management, since its specification may indicate the type of spoken utterance expected from the user at each stage of the dialogue, up to the precise speech recognition model and parameter values to use, and the generation of the system utterance, from natural language generation to speech synthesis or audio recordings.

Our dialogue design tool offers to the SDS developers a graphical abstraction of the dialogue logic, sometimes also named the call flow. Thanks to a dynamic VoiceXML generation functionality, our dialogue design tool brings the SDS developers the guarantee that VUI-completeness at the design level automatically implies a similar completeness at the implementation level. During maintenance, If the SDS developers modify a specific part of the dialogue design, the tool guarantees that solely the corresponding code is impacted. This guarantee impacts positively VUI-completeness, reliability, and development cost.

Figure 1 presents the design of a typical VoiceXML page. This page is used when the system

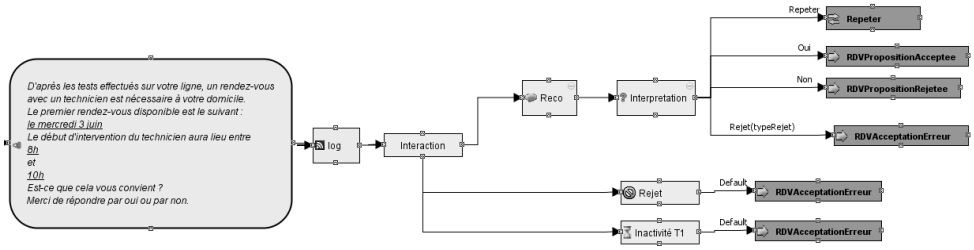


FIGURE 1 – 1013+ design excerpt : the system asks the user to confirm an appointment slot

asks the user to accept an appointment time slot. It first begins with a prompt box mixing static and dynamic prompts (the dynamic parts are underlined and realised by service-specific java code). A log box is then used some contextual session variables. Then, an interaction box is used to model the system reaction to the user behaviour : on the lower part of the Figure, we program the reaction to user inactivity or recognizer misunderstanding. In the upper part, we use a recognition box followed by a Natural Language Understanding (NLU), and we program the different output classes : repeat, yes, no and not understood. Each output is linked to a transition box, which indicates which VoiceXML page the service should call next.

3.2 Monitoring Functionalities inside the Design Tool

While researchers are focused on measuring the progress they incrementally reach, industry engineers have to deal with SDS tuning and upgrade. Their first dialogue evaluation KPI is task completion also called the automation rate because a SDS is deployed to automate specifically selected tasks. Most of the time, task completion is estimated thanks to the KPI. The KPI are difficult to exhaustively list and classify. Some are related to system measures, others are obtained thanks to dialogue annotations and the last ones are collected from users through questionnaires.

Some studies (Abella et al., 2004) investigated graphical monitoring tools. The corpus to visualise is a set of dialogue logs. The tool aims at revealing how the system transits between its possible states. As a dialogue system is too complex to enumerate all its possible states, the dialogue logs are regarded as a set of variables that evolve during time and the tool proposes to make a projection on a subset of these variables. This way, the generated graphs can either display the call flow, how the different steps are reached and where they lead, or display how different variables, as the number of errors evolve. This is mainly a tool for understanding how the users behave, because it has no direct connection with the way how the system was built. As consequence to this, it does not help to diagnose how to make it better. In other words, it does evaluate the system but does not meet one of our goal : the convergence between design and evaluation.

On the opposite, our graphical design tool provides an innovative functionality : local KPI projection into the original dialogue design thanks to an extensive logging. A large part of the KPI are automatically computed and displayed. As a consequence, it is possible to display percentage of which responses the system recognised, the users actually gave, and see how these numbers match the various KPI. It is one example among the numerous analysis views this graphical tool can provide.

3.3 Insertion of Alternatives

The 1013+ service has been used to test three kinds of design alternatives. The first kind is a strategy alternative : the service can choose between offering an appointment time slot to the client, or asking her for a time slot. This decision defines whether the next dialogue step will be system-initiative or user-initiative. The second kind is a speaking style alternative : the service can either be personified by using the “I” pronoun, adopt a corporate style by using the “We” pronoun, or speak in an impersonal style by using the passive mode. The third kind is a Text-To-Speech alternative : the service can use a different wording or prosody for a given sentence.

Figure 2 displays a monitoring view of an interaction implementation with alternatives. The recognition rate is the projected KPI on the graph at each branch. Other performance indicators are dis-

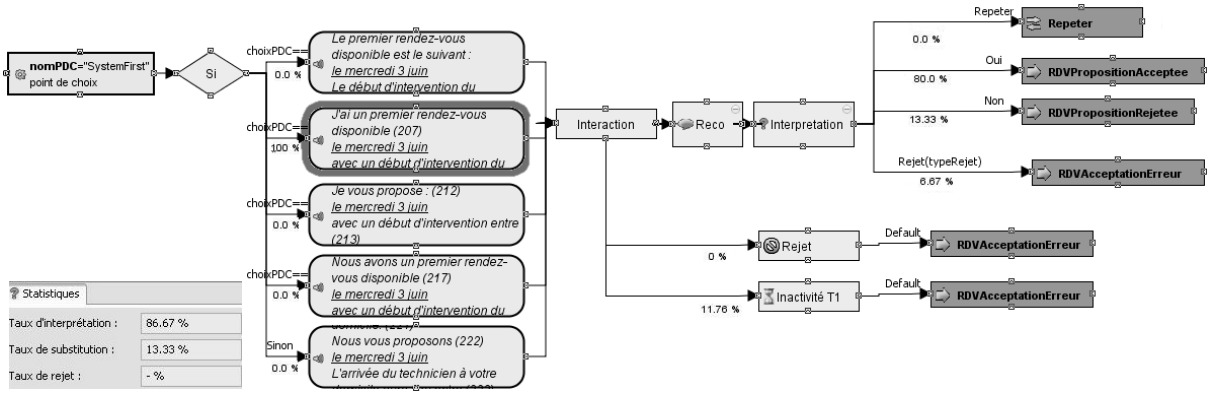


FIGURE 2 – Some user experience feedbacks related to a selected prompt alternative.

played at the bottom of the window : here, it is the actual rate of correct semantic decoding, the semantic substitution rate, and the semantic rejection rate. The selection of the highlighted box conditions the displayed logs.

Our design tool also provides a multivariate testing functionality. This method consists in testing multiple alternatives and selecting the best one on a fixed set of predetermined criteria. Regarding the VUI-completeness, presenting the complete automaton to the SDS developers is acceptable, as long as they can inspect and control every branch of the design. In general, they even come up with several competing designs or points of choice, which can only be properly selected from in a statistical manner. The ability to compare all the dialogue design alternatives in the same test-field is a major factor to boost up SDS enhancement by drastically reducing the time needed. When we were developing the current 1013+ version, we have been able to develop the 5 main alternatives in less than a month, where it had taken a month and a half for a unique alternative in previous versions. It brings a statistical relevance in the causal link between the tested alternatives and the differences in performance measures, because it ensures a good random input space coverage.

The KPI graphical projection into the dialogue design covers the dialogue alternatives : KPI computation just needs to be conditioned by the alternatives. Figure 2 illustrates the merge of several system prompt alternatives inside a single design. It represents the prompt alternatives the system can choose when proposing an appointment time slot. An action block informs the Learning Manager about the current dialogue state and available dialogue alternatives. An “If” block then activates the prompt alternative corresponding to a local variable “choixPDC” filled by the Learning Manager. The rest of the design is identical to the design presented in Figure 1.

The displayed KPI are conditioned by the selected alternative (here, the second wording circled in bold grey). ODDS then indicates how the dialogue call flow is breakdown into the different alternatives. As we have here conditioned the displayed information by the second alternative, this alternative receives 100% of the calls displayed, when the other alternatives are not used. We can then see the different outcomes for the selected alternative : the customer answer have lead to a timeout of the recognition in 11.78% of the cases, and amongst the recognised sentences, 80% were an agreement, 13.33% were a reject, and 6.67% were not understood.

On the bottom-left part, one can display more specific KPI, such as good interpretation rate, substitution rate, and reject rate. These KPI are computed after the collected logs have been manually annotated, which remains an indispensable process to monitor and improve the recognition and NLU quality, and thus the overall service quality.

Conditioning on another alternative would have immediately led to different results, and someway, embedding the user experience feedback inside the dialogue design forms a new material to touch and feel : the SDS developers can now sculpt a unique reactive material which contains the design and the KPI measures distribution. By looking at the influence of each alternative on the KPI when graphically selecting the alternatives, the SDS developers are given a reliable means to understand how to improve the system.

3.4 Reassessing Dialogue Evaluation

The traditional approaches to dialogue evaluation attempt to measure how best the SDS is adapted to the users. We remind that each interaction between the user and the SDS appears to be a unique performance. First, each new dialogue is co-built in a unique way according to both the person-specific abilities of the user and the possibilities of the SDS. Second, the user adapts very quickly to new situations and accordingly changes her practices. The traditional approaches to dialogue evaluation are eventually based on the fragile reference frame of the user, not reliable enough for a scientific and an industrial approach of the spoken dialogue field, mostly because of the inability to get statistical call volumes for all the dialogue alternatives.

This suggests for a shift in the reference frame used for dialogue evaluation : instead of trying to measure the adequacy between the SDS and the user in the user's reference frame, one can measure the adequacy between the user and the SDS in the design reference frame composed by the dialogue logic, the KPI and their expected values. Taking the design as the reference allows reassessing the dialogue evaluation. The proposed basis for dialogue evaluation is reliable for the SDS developers because it is both stable and entirely under control. Deviations from the predicted situations are directly translated into anomalous values of measurable KPI that raise alerts. These automatically computable alerts warn the SDS developers about the presence of issues in their dialogue design.

4 Dialogue design learning

As presented in previous Section, the alternative insertion is an enabler for the dialogue system analysis tools. It provides the SDS developers with a novel call flow visualisation experience. The further step to this approach is to automate at least a part of those analyses and improvements with learning capabilities.

4.1 Constraints

The objective is to automatically choose online the best alternative among those proposed in the design tool, and to report this choice to the SDS developers via the monitoring functionalities that are integrated to the design tool. This approach differs from the classical reinforcement learning methods used in the dialogue literature, which make their decisions at the dialogue turn level.

We use a technique from a previous work (Laroche et al., 2009). It does not need to declare the reachable states : they are automatically created when reached. This is also a parameter-free algorithm, which is very important when we consider that most dialogue application developers are not familiar with reinforcement learning theory. We keep the developer focussed on its main task. The two additional tasks required for the reinforcement learning are to define the variable set on which the alternative choice should depend, and to implement a reward function based on the expected evaluation of the task completion, in order to get a fully automated optimisation with an online evaluation. The dialogue system automatic evaluation is a large problem that goes beyond the scope of this paper. However, sometimes, the dialogue application enables to have an explicit validation from the user. For instance, in an appointment scheduling application, the user is required to explicitly confirm the schedule he was proposed. This user performative act completes the task and provides a reliable automatic evaluation.

4.2 Learning and Monitoring Synergy in the Design Optimisation

The learning algorithm and the SDS developers are two actors on the same object : the dialogue system. But, they work at a different time space. The learning algorithm updates its policy after each dialogue while the SDS developers monitor the system behaviour more occasionally. The same kind of opposition can be made on the action space of those actors. The learning algorithm can only change its policy among a limited amount of alternatives, while the SDS developers can make deeper changes, such as implementing a new dialogue branch, adding new alternatives, new alternative points, removing alternatives, etc...

Last but not least, their sight ranges vary a lot too. The learning algorithm is concentrated on the alternative sets and automatic evaluation and ignores the rest, while the SDS developers can apprehend the dialogue application as a whole, as a system or as a service. They can also have access to additional evaluations through annotations, or user subjective evaluations.

These functionality differences make their respective roles complementary. The SDS developers

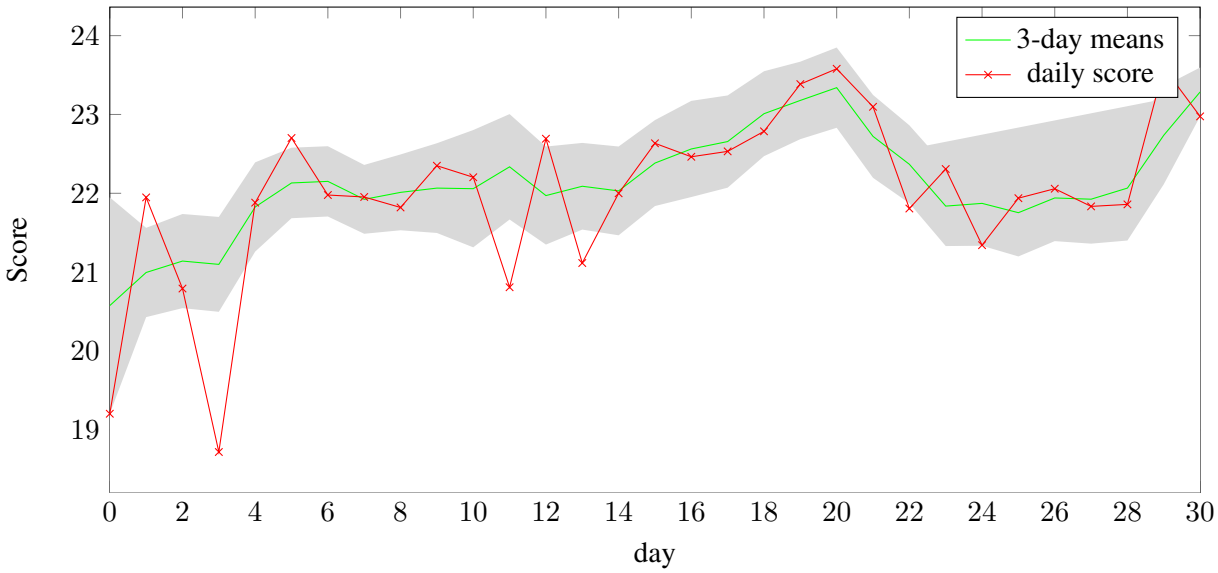


FIGURE 3 – Evolution of the system score : baseline

have the responsibility for the whole application and the macro-strategic changes while the learning manager holds the real-time optimisation.

4.3 Control vs. Automation : the Trusting Threshold

As argued by Pieraccini and Huerta (Pieraccini and Huerta, 2005), finite state machine applied to dialogue management does not restrict the dialogue model to strictly directed dialogues. Finite state machines are easily extensible to powerful and flexible dialogue models. Our dialogue design tool offers various extensions : dialogue modules, hierarchical design, arbitrary function invocation at any point of the design, conditional statements to split the flow in different paths. All those extensions allow designing any topology of the finite state machine required to handle complex dialogue models like mixed-initiative interaction. Dialogue model is not the point where research and industry fail to converge.

The divergence point concerns the control aspect of VUI-completeness versus the automation of the dialogue design. As pointed out by recent works (Paek and Pieraccini, 2008), MDP-based dialogue management aiming at automating the whole dialogue design is rejected by the SDS developers. Even more adaptive, it is seen as an uncontrollable black box sensitive to the tuning process. The SDS developers do not rely on systems that dynamically build their dialogue logic without a sufficient degree of monitoring and control.

Williams (Williams, 2008) has made a substantial effort to meet this industrial requirement. His system is a hybridisation of a conventional dialogue system following an industrial process, with a POMDP decision module, which is a MDP-based approach to dialogue management enhanced with dialogue state abstractions to model uncertainties. The responsibilities of each part of the system are shared as follows : the conventional system elects several candidate dialogue moves and the POMDP decision module selects the most competitive one. This is a great step towards industry because the dialogue move chosen by the POMDP module has been first controlled by the conventional system design. Nevertheless, the so-built hybrid system is still not fully compliant with the industrial constraints for the following reasons.

First, contrary to our approach, the SDS developer is called upon specific skills that cannot be demanded to a developer (modeling and tuning a (PO)MDP). This is a no-go for further integration in an industrial process.

Second, such a predictive module is not self-explanatory. Although the SDS developers have the control on the possible behaviour presented to the POMDP decision module, they are given no clue to understand how the choices are made. In fact, a learnt feature can never be exported to another context. At the opposite, our approach allows us to learn at the design level and consequently to report in the automaton the optimisation. The learning results are therefore understandable, ana-

lysable and replicable on a larger scale, in a way similar to classical ergonomics guidelines (but statistically proved).

4.4 Learning results on the 1013+ service

In the 1013+ service, our experiments have focused on the appointment scheduling domain. We have chosen to integrate the following rewards in the service : each time a user successfully manages to get an appointment, the system is given a $+30$ reward. If the system is unable to provide an appointment, but manages to transfer the user to a human operator, the system is given a $+10$ (a "resit"). Last, if the user hangs up, the system is not given any positive reward. Every time the system does not hear nor understand the user, it is given a penalty of 1.

In the beginning of the experiment, when the system is still using a random policy, the completion rate is as low as 51%, and the transfer rate is around 36%. When the system has learned its optimal policy, the completion rate raises up to 70%, with a transfer rate around 20%. In our experiment, the system has learned to favour an impersonal speaking style (passive mode) and it prefers proposing appointment time slots rather than asking the user to make a proposition (the later case leading to lot of "in private" user talks and hesitations, and worse recognition performance).

Figure 3 shows the evolution of the mean dialogue score during the first month. Each server have its own Learning Manager database, and optimises separately. This is a welcome feature, as each server can address a different part of the user population, which is a frequent operational requirement.

The dialogue score drawn on Figure 3 is computed by averaging the mean dialogue score per server. The crossed line represents the daily mean dialogue score. The normal line represents the 3-day smoothed dialogue mean score. The grayed area represents the 95% confidence interval. During this first month of commercial exploitation, one can notice two major trends : at first, the dialogue score is gradually increasing until day 20, then the performances noticeably drops, before rising up again. It turns out that new servers were introduced on day 20, which had to learn the optimal dialogue policy. Ultimately (on the second month), they converge to the same solution as the first servers.

5 Conclusion

5.1 A New Basis for Trusting Automatic Learning

This paper presents an original dialogue design tool that mixes dialogue design and dialogue evaluation in the same graphical interface. The design paradigm supported by the tool leads the SDS developers to predict value ranges of local KPI while designing the dialogue logic. It results a new evaluation paradigm using the system design as the reference and trying to measure deviations between the predicted and the measured values of the designed local KPI. The SDS developers rely on the tool to fulfil the VUI-completeness principle. Classically applied to dialogue design, the tool enables its application to the dialogue evaluation, leading to the comparison of dialogue design alternatives.

This places the SDS developers in a dialogue design improvement cycle close to the reinforcement learning decision process. Moreover, the inspector offered by the user experience feedback functionality allows the SDS developers to understand, analyse and generalize all the decisions among the dialogue design alternatives. Combining the learning framework and the design tool guarantees the SDS developers keep control of the system. It preserves VUI-completeness and opens the way to a reliable learning based dialogue management.

5.2 Implementation

This approach to learning led us to deploy in October 2009 the first commercial spoken dialogue system with online learning. The system's task is to schedule an appointment between the customer and a technician. This service receives approximately $8,000$ calls every month. At the time those lines are written, we are already in a virtuous circle of removing low-rated alternatives and replacing them with new ones, based on what the system learnt and what the designer understands from the data.

5.3 Future Work

On a social studies side, we are interested in collaborations to test advanced dialogue strategies and/or information presentation via generation. Indeed, we consider our system as a good opportunity for large scope experiments.

6 Acknowledgements

This research has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 216594 (CLASSIC project : www.classic-project.org).

References

A. Abella, J.H. Wright, and A.L. Gorin. 2004. Dialog trajectory analysis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 441–444, May.

R.E. Bellman. 1957. A markovian decision process. *Journal of Mathematics and Mechanics*, 6 :679–684.

J. Bos, E. Klein, O. Lemon, and T. Oka. 2003. Dipper : Description and formalisation of an information-state update dialogue system architecture.

George Ferguson and James F. Allen. 1998. Trips : An integrated intelligent problem-solving assistant. In *In Proc. 15th Nat. Conf. AI*, pages 567–572. AAAI Press.

R. Laroche, G. Putois, P. Bretier, and B. Bouchon-Meunier. 2009. Hybridisation of expertise and reinforcement learning in dialogue systems. In *Proceedings of Interspeech. Special Session : Machine Learning for Adaptivity in Spoken Dialogue*, Brighton (United Knigdom), September.

O. Lemon and O. Pietquin. 2007. Machine learning for spoken dialogue systems. In *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)*, pages 2685–2688, August.

M. F. McTear. 2004. *Spoken Dialogue Technology : Toward the Conversational User Interface*. Springer, August.

T. Paek and R. Pieraccini. 2008. Automating spoken dialogue management design using machine learning : An industry perspective. *Speech Communication*, 50 :716–729.

T. Paek. 2007. Toward evaluation that leads to best practices : Reconciling dialog evaluation in research and industry. In *Proceedings of the Workshop on Bridging the Gap : Academic and Industrial Research in Dialog Technologies*, pages 40–47, Rochester, NY, April. Association for Computational Linguistics.

R. Pieraccini and J. Huerta. 2005. Where do we go from here ? research and commercial spoken dialog systems. In Laila Dybkjaer and Wolfgang Minker, editors, *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, pages 1–10.

R. Pieraccini, S. Caskey, K. Dayanidhi, B. Carpenter, and M. Phillips. 2001. Etude, a recursive dialog manager with embedded user interface patterns. In *Automatic Speech Recognition and Understanding, 2001 IEEE Workshop on*, pages 244–247.

M. D. Sadek, P. Bretier, and F. Panaget. 1997. Artimis : Natural dialogue meets rational agency. In *in Proceedings of IJCAI-97*, pages 1030–1035. Morgan Kaufmann.

J. D. Williams. 2008. The best of both worlds : Unifying conventional dialog systems and POMDPs. In *International Conference on Speech and Language Processing*.