Optimizing Spoken Dialogue Management from Data Corpora with Fitted Value Iteration

Senthilkumar Chandramohan, Matthieu Geist, Olivier Pietquin

IMS research group, Supélec, Metz Campus, France

{senthilkumar.chandramohan, matthieu.geist, olivier.pietquin} @supelec.fr

Abstract

In recent years machine learning approaches have been proposed for dialogue management optimization in spoken dialogue systems. It is customary to cast the dialogue management problem into a Markov Decision Process (MDP) and to find the associated optimal policy using Reinforcement Learning (RL) algorithms. Yet, the dialogue state space is usually very large (even infinite) and standard RL algorithms fail to handle it. In this paper we explore the possibility of using a generalization framework for dialogue management which is a particular fitted value iteration algorithm (namely fitted-Q iteration). We show that fitted-Q, when applied to continuous state space dialogue management problems, can generalize well and makes efficient use of samples to learn the approximate optimal state-action value function. Our experimental results show that fitted-Q performs significantly better than the hand-coded policy and relatively better than the policy learned using least square policy iteration (LSPI), another generalization algorithm.

Index Terms: Spoken Dialogue Systems, Dialogue Management, Reinforcement Learning

1. Introduction

Spoken Dialogue Systems (SDS) are systems having the ability to interact with human beings using speech as the communication medium. They usually aim at accomplishing a specific task such as town information access [10], flight booking [23], *etc*. The dialogue management module of the SDS is responsible for navigating the dialogue system so that it can accomplish the designated task. Given the complexity of dialogue systems (due to uncertainty in capturing the user inputs, stochastic user behaviour, background noise), it remains a challenge and an expert job to build a robust and adaptive dialogue manager.

In recent years, several machine learning approaches to dialogue management have been proposed to deal with the complexity of this task [11, 17, 21, 15, 23] and automatically learn a management strategy. Dialogue management is indeed a sequential decision making problem and thus can be cast succinctly into a Markov Decision Process (MDP) [3, 11] or a Partially Observable Markov Decision Process (POMDP) [1, 23]. Reinforcement learning (RL) [22] can therefore be applied to learn the optimal management policy. Yet standard RL algorithms require a large amount of data to converge. In the spoken dialogue domain, collecting and annotating data (or using a Wizard-of-Oz setup [16]) is very time consuming. To alleviate this data sparsity problem, user modeling and dialogue simulation have received a great interest in the last decade [12, 20, 14]. But this technique introduces new sources of modeling errors and the effect of the user simulation method on the learned strategy is still not very well known [19].

Most of state-of-the-art RL approaches to dialogue management use Temporal Difference (TD)-based RL algorithms (most often the SARSA or Q-learning algorithm) to optimize the dialogue policy. These algorithms have been developed for systems having a limited number of states. When MDP with large number of states or continuous state representations are encountered, most of these algorithms are inefficient if applicable and fail to determine a good policy in previously unseen situations. This is especially true with regard to SDS where the state space of the dialogue problem is often continuous (due to the introduction of continuous speech recognition and understanding confidence levels in the state space). The primary motivation for this work is to explore the possibility of using approximate dynamic programming and especially a Fitted Value Iteration (FVI) algorithm [4, 18] to approximate the continuous state-action value function (or Q-function) associated to the optimal policy and thus infer the best dialogue policy.

Fitted-Q, a specific FVI algorithm, aims at approximating the optimal state-action value (or Q-) function for an MDP from a fixed set of data samples. This class of RL algorithms is known to make effective use of samples and to show good generalization properties. The main advantage of using fitted-Q is twin-folded: (i) it can approximate the underlying optimal policy directly from available corpora (ii) by choosing a suitable regression technique it can exploit the generalization capability and sample efficiency of any supervised-learning-based function approximation method. Another significant advantage of fitted-Q is that the algorithm has no tunable parameters.

Very limited work has been done in the field of approximate dynamic programming applied to SDS management. Of all the recent work in the field of machine learning for SDS we believe that there are two closely related approaches, (i) generalization of state-action value function in large state space dialogue management problems using hybrid learning [7] and (ii) using least-squares policy iteration (LSPI) [8] for dialogue management [13]. The layout of the paper is as follows. Section 2 gives an overview on using reinforcement learning for spoken dialogue management and Section 3 provides an outline of the fitted-Q algorithm. Description of dialogue management using fitted-Q along with the experimental set-up is outlined in Section 4, following which in Section 5 are stated our conclusion based on experimental results and then outline the future direction of the work described here.

2. RL for dialogue management

2.1. Dialogue management as a decision making problem

Spoken dialogue management can be seen as a sequential decision making problem. Here the dialogue manager (decision maker) has to select and perform an action (system dialogue

act) from a set of possible actions. The action selection is based on the current state of the dialogue manager, frequently termed as the dialogue state. For example in the case of a restaurant information system at any given time there can be more than one possible system action such as ask-restaurant-cuisine, ask-restaurant-price-range, ask-restaurant-location etc. One compact and efficient way of representing the dialogue state is given by the Information State paradigm as suggested in [9]. The information state or simply the dialogue state is a succinct representation of the history of system acts and its subsequent user responses (user acts).

A dialogue strategy is therefore a mapping between dialogue states and dialogue acts. The optimal strategy is the one that maximizes some reward function which is usually defined as the contribution of each actions to the user's satisfaction [21]. This subjective reward is usually estimated from a linear combination of objective measures (dialogue duration, number of ASR errors, *etc*.).

2.2. Markov Decision Process

Markov Decision Processes (MDP) [2] have been used traditionally as an effective framework for solving sequential decision making problems. An MDP is defined as a tuple $\{S, A, R, T, \gamma\}$, where S is the set of all possible states, A is the set of actions, R is the reward function, T is the set of Markovian transition probabilities and γ is the discount factor. A mapping from $s \in S$ to $a \in A$ is termed as a policy π for the MDP. Given the policy π the state value function ($V^\pi : S \rightarrow \mathbb{R}$) is defined as the expected discounted sum of rewards that can be obtained by an agent over the infinite horizon starting from state s and following the policy π: $V^\pi(s) = E[\sum_{k=0}^{\infty} \gamma^k r_k | s_0 = s, \pi]$. The state-action value (or Q-) function ($Q^\pi : S \times A \rightarrow \mathbb{R}$) adds a degree of freedom for the choice of the first performed action: $Q^\pi(s,a) = E[\sum_{k=0}^{\infty} \gamma^k r_k | s_0 = s, a_0 = a\pi]$. The optimal policy maximizes the value of each state (or state-action pair): $\pi^* = \text{argmax}_\pi V^\pi = \text{argmax}_\pi Q^\pi$.

The optimal policy is greedy respectively to the optimal state-action value function Q^*: $\pi^*(s) = \text{argmax}_{a \in A} Q^*(s,a)$. Finding the optimal policy therefore reduces to finding the optimal Q-function, which is the solution of the so-called Bellman optimality equation (or equivalently the fixed point of the Bellman optimality operator T^*):

$$Q^*(s,a) = E_{s'|s,a}[R(s,a,s') + \gamma \max_{b \in A} Q^*(s',b)] \quad (1)$$

$$\Leftrightarrow Q^* = T^* Q^* \quad (2)$$

This T^* operator can be shown to be a contraction [2] and therefore admits an unique fixed point. The Banach theorem provides an incremental procedure to estimate it, known as *Value Iteration* in the Dynamic Programming literature:

$$\hat{Q}_i^* = T^* \hat{Q}_{i-1}^* \quad (3)$$

3. Fitted value iteration

Value iteration is therefore an algorithm which allows computing the optimal Q-function. However, it assumes that the state-action value function admits an exact representation, which is not possible when the state space is too large. In such a case, it is convenient to adopt a parametric representation. Let $\theta \in \mathbb{R}^p$ represent the parameter vector and $\hat{Q}_\theta$ the approximate Q-function belonging to the hypothesis space $\mathcal{H}$ (the functional space spanned by parameters). The goal is therefore to compute a good approximation $\hat{Q}_\theta$ of Q^*. To do so, the fitted value iteration (FVI) class of algorithms [4, 18] adopts an approximate value iteration scheme. However, the image of $\hat{Q}_\theta$ through the Bellman operator does not necessarily lies in $\mathcal{H}$, it should be projected onto the hypothesis space. By noting Π the projection operator, defined for any function $f : S \times A \rightarrow \mathbb{R}$ as $\Pi f = \text{argmin}_{\hat{Q}_\theta \in \mathcal{H}} \|f - \hat{Q}_\theta\|^2$, this leads to fining the parameter vector θ satisfying:

$$\hat{Q}_\theta^* = \Pi T^* \hat{Q}_\theta^* \quad (4)$$

Fitted-Q, which is a specific FVI algorithm, assumes that the composed ΠT^* operator is a contraction and therefore admits an unique fixed point, which is searched for through the following classic iterative scheme: $\hat{Q}_{\theta_i} = \Pi T^* \hat{Q}_{\theta_{i-1}}$. However, the transitions probabilities T and reward function R are usually not known (and especially in the spoken dialogue system domain), therefore a *sampled* Bellman optimality operator $\hat{T}^*$ is considered instead. For a sample transition (s_i, a_i, r_i, s_i'), it is defined as:

$$\hat{T}^* Q(s_i, a_i) = r_i + \gamma \max_{a \in A} Q(s_i', a) \quad (5)$$

This provides a general fitted-Q algorithm (θ_0 being chosen by the user):

$$\hat{Q}_{\theta_i} = \Pi \hat{T}^* \hat{Q}_{\theta_{i-1}} \quad (6)$$

Fitted-Q can then be specialized by choosing how $\hat{T}^* \hat{Q}_{\theta_{i-1}}$ is projected onto the hypothesis space (that is by choosing a supervised learning algorithm matching $\hat{Q}_{\theta_i}(s_j, a_j)$ to $\hat{T}^* \hat{Q}_{\theta_{i-1}}(s_j, a_j) = r_j + \gamma \max_{a \in A} \hat{Q}_{\theta_{i-1}}(s_j', a)$ for each training transitions (s_j, a_j, r_j, s_j')).

In this paper, a linear parametrization $\hat{Q}_\theta(s,a) = \sum_i \theta_i \phi_i(s,a) = \theta^T \phi(s,a)$ is adopted, with $\phi(s,a)$ the set of p basis functions. Assume also that a training basis $\{(s_j, a_j, r_j, s_j')_{1 \leq j \leq N}\}$ is available. The projection is done thanks to the solution of the corresponding least-squares optimization problem (we note $\phi(s_j, a_j) = \phi_j$):

$$\theta_i = \underset{\theta \in \mathbb{R}^p}{\text{argmin}} \sum_{j=1}^{N} (\hat{T}^* \hat{Q}_{\theta_{i-1}}(s_j, a_j) - \hat{Q}_\theta(s_j, a_j))^2 \quad (7)$$

$$= \underset{\theta \in \mathbb{R}^p}{\text{argmin}} \sum_{j=1}^{N} (r_j + \gamma \max_{a \in A} (\theta_{i-1}^T \phi(s_j', a)) - \theta^T \phi_j)^2 \quad (8)$$

$$\theta_i = (\sum_{j=1}^{N} \phi_j \phi_j^T)^{-1} \sum_{j=1}^{N} \phi_j (r_j + \gamma \max_{a \in A} (\theta_{i-1}^T \phi(s_j', a))) \quad (9)$$

Equation (9) defines an iteration of the proposed linear least-squares-based fitted-Q algorithm. An initial parameter vector θ_0 should be chosen, and iterations are stopped when some criterion is met (maximum number of iterations or small difference between two consecutive parameter vector estimates). Assume that there are M iterations, the optimal policy is estimated as $\hat{\pi}^*(s) = \text{argmax}_{a \in A} \hat{Q}_{\theta_M}(s,a)$.

Notice that the inverted matrix $\left(\sum_{j=1}^{N} \phi_j \phi_j^T\right)^{-1}$ is the same for each iteration (only targets $r_j + \gamma \max_{a \in A} (\theta_{i-1}^T \phi(s_j', a))$ depend on predeceasing estimate), therefore the complexity per iteration is in $O(p^2)$. This is much lower than LSPI [13] which is in $O(p^3)$ per iteration. LSPI is an approximate policy iteration algorithm [22], whereas fitted-Q is an approximate value iteration algorithm. It is also known for its efficient use of samples and its off-policy aspect.

4. Experimental set-up and results

In this section we give an outline of our experimental setup. To begin with we first describe our MDP-SDS and then describe the source of our training data. We then describe the representation of our Q-function following which we outline how fitted-Q can be used for dialogue management.

4.1. Dialogue task and RL parameters

Our dialogue system is a task-oriented, form-filling dialogue system in the tourist information domain similar to the one studied in [10]. The goal of our system is to give information about restaurants in the city based on specific user preferences. We primarily have three slots in our dialogue problem (i) location of the restaurant, (ii) cuisine of the restaurant and (iii) price-range of the restaurant. Our dialogue state has three continuous components ranging from 0 to 1, each representing the average of filling and confirmation confidence of their respective slots. Our MDP SDS has the following 13 actions, Ask-slot (3 actions), Explicit-confirm (3 actions), Implicit-confirm and Ask-slot value (6 actions) and finally Close-dialogue (1 action). Since we intend to perform dialogue policy optimization in a model free set-up we do not need the set of transition probabilities T. The discount factor was set 0.95 in order to encourage delayed rewards [22] and also to induce an implicit penalty for the length of the dialogue episode. The reward function R of our MDP-SDS is presented as follows: every correct slot filling is awarded 25, every incorrect slot filling is awarded -75 and every empty slot filling is awarded -300.

4.2. Dialogue corpora for policy optimization

In order to perform fitted-Q for dialogue management we need a dialogue corpora which represents the problem space. As in case of any batch learning method the samples used for learning should be chosen (if they can be chosen) in such a way that they span across the problem space. In this experiment, a user simulation technique was used to generate the data corpora. This way, the sensibility of the method to the size of the training dataset could be analysed (available human-dialogue corpora are limited in size). The user simulator was plugged to the DIPPER [10] dialogue management system to generate dialogue samples. In order to generate data, the dialogue manager strategy was jointly based on a simple hand-coded policy (which aims to fill all the slots before closing the dialogue episode) and random action selection. Randomly selected system acts are used with probability ϵ and hand-coded policy selected system acts are used with probability (1-ϵ). During our data generation process the ϵ value was set to 0.9. Rather than using a fully random policy we used an ϵ-greedy policy to ensure that the problem space is well sampled and in the same time at least few episodes have successful completion of task compared to a totally random policy. We ran 56,485 episodes between the policy learner and an unigram user simulation, using the ϵ-greedy policy (of which 65% are successful task completion episodes) and collected 39,3896 dialogue turns (state transitions).

4.3. Linear representation of Q-function

The samples generated were then used for dialogue policy optimization using fitted-Q. We used Radial basis function (RBF) for representing our Q-function (*i.e,* the features ϕ_i discussed in section are Gaussian functions centred in μ_i and with the same standard deviation $\sigma_i = \sigma$). Our RBF network had 3 Gaussians for each dimension in the state vector and considering that we have 13 actions, in total we used 351 (*i.e,* $3^3 \times 13$) features for approximating the Q-function. Gaussians were centred at ${}_m u_i$ = 0.0, 0.5, 1.0 in every dimension with a standard deviation $\sigma_i = \sigma = 0.25$ (data are normalized). We used the least-squares regression technique as to fit the Q-function (*i.e,* optimize the weights θ). Our stopping criteria was based on comparison between L_1 norm of succeeding weights and a threshold ξ which was set to 10^{-2} *i.e,* convergence if $\sum_i \left(|\theta_i^n - \theta_i^{n-1}|^1\right) < \xi$, where n is the iteration number.

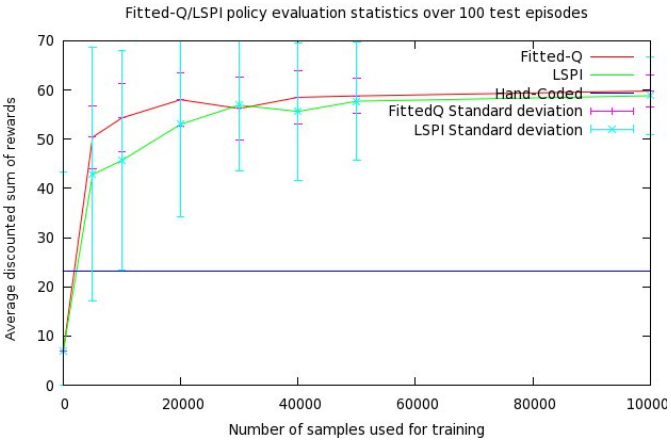

Figure 1: *Policy evaluation statistics*

4.4. Evaluation of learned policy

Our first aim was to evaluate the sample efficiency of least-squares fitted-Q when applied to dialogue management problems. So we ran a set of (learning) fitted-Q iterations using different numbers of training samples (one sample is a dialogue turn, that is a state transition {s,a,r,s'}). The number of samples used for training ranged from 4.10^3 to 100.10^3 samples (no convergence of weights was observed during training for number of samples below 4.10^3). The policies were then tested using a unigram user simulation and the DIPPER dialogue management framework. We also ran test episodes using the ϵ-greedy policy used for data generation (initial policy corresponding to zero samples) and the hand-coded policy. Figure 1 shows the average discounted sum of rewards and associated standard deviation of every policy tested over 100 dialogue episodes using a simulated user and DIPPER.

4.5. Analysis of evaluation results

In order to have an unified evaluation for fitted-Q and LSPI, we used the same set of basis functions for LSPI. We learned the optimal dialogue policies using LSPI with different numbers of training samples and evaluated the resulting policies as explained above. Evaluation results of policies learned using LSPI are also presented in Figure 1.

Our experimental results show that the dialogue policy learned using fitted-Q performs significantly better than the hand-coded policy and the epsilon greed policy (initial policy for 0 samples in Figure 1) used for data collection. Most importantly it can be observed from the figure that the fitted-Q based dialogue management is sample efficient and needs only few thousand samples (recall that a sample is a dialogue turn and not a dialogue episode) to learn fairly good policies, thus ex-

hibiting a possibility to learn a good policy directly from very limited amount of dialogue corpus. We believe this is a significant improvement when compared to the corpora requirement for dialogue management using other RL algorithms such as SARSA.

Our results also show that the efficiency of the dialogue policies learned using fitted-Q is marginally better when compared to the policies learned using LSPI in terms of average discounted sum of rewards. Moreover, it can be observed from Figure 1 that LSPI policies have a larger standard deviation (with regard to expected discounted sum of rewards) compared to fitted-Q policies, irrespective of the number of samples used for training. Since the policies learned using fitted-Q have relatively less variations we believe that these policies are more stable and thus more reliable when compared to LSPI policies.

From a computational complexity point of view, that fitted-Q and LSPI roughly need the same number of iterations before that the stopping criterion is met. However, reminding that the proposed fitted-Q complexity is $O(p)^2$ per iteration whereas LSPI complexity is $O(p^3)$ per iteration, fitted-Q is computationally less intensive.

5. Conclusion and future works

Our primary motivation for using fitted-Q for spoken dialogue management is to exploit the sample efficiency and good generalization capability of least-squares fitted-Q for continuous or large state space dialogue problems. From experimental results it can be shown that spoken dialogue management using fitted-Q is a sample efficient approach for learning not only optimal but also stable dialogue policies compared with other RL algorithms such as LSPI and SARSA.

The linear function representation used in the work explained above used hand-selected features (although quite generic) and we believe that one interesting direction of work will be to explore the possibilities of using automatic feature selection methods as in [13]. The kernel recursive least-squares algorithm [5] is probably a good candidate method to do so and can easily be integrated in the fitted-Q paradigm. One other interesting direction of work will be to learn an initial policy using LSPI or fitted-Q and then explore the possibilities of improving the policy using sample-efficient online RL algorithms such as KTD [6].

6. Acknowledgements

The work presented here is part of an ongoing research for CLASSiC project (Grant No. 216594, www.classic-project.org) funded by the European Commission's 7th Framework Programme (FP7).

7. References

[1] K. J. Astrom. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 1965.

[2] R. Bellman. *Dynamic Programming*. Dover Publications, sixth edition, 1957.

[3] R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 1957.

[4] R. Bellman and S. Dreyfus. Functional approximation and dynamic programming. *Mathematical Tables and Other Aids to Computation*, 1959.

[5] Y. Engel, S. Mannor, and R. Meir. The Kernel Recursive Least Squares Algorithm. *IEEE Transactions on Signal Processing*, 2004.

[6] M. Geist, O. Pietquin, and G. Fricout. Kalman temporal differences: the deterministic case, 2009.

[7] J. Henderson, O. Lemon, and K. Georgila. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics*, 2008.

[8] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 2003.

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

[10] 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, 2006.

[11] E. Levin and R. Pieraccini. Using markov decision process for learning dialogue strategies, 1998.

[12] 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*, 2000.

[13] L. Li, S. Balakrishnan, and J. Williams. Reinforcement Learning for Dialog Management using Least-Squares Policy Iteration and Fast Feature Selection, 2009.

[14] O. Pietquin. Consistent goal-directed user model for realistic man-machine task-oriented spoken dialogue simulation, July 2006.

[15] O. Pietquin and T. Dutoit. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech & Language Processing*, 2006.

[16] V. Rieser. *Bootstrapping Reinforcement Learning-based Dialogue Strategies from Wizard-of-Oz data*. PhD thesis, Saarland University, Dpt of Computational Linguistics, July 2008.

[17] N. Roy, J. Pineau, and S. Thrun. Spoken dialogue management using probabilistic reasoning, 2000.

[18] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development*, 1959.

[19] J. Schatzmann, M. N. Stuttle, K. Weilhammer, and S. Young. Effects of the user model on simulation-based learning of dialogue strategies, December 2005.

[20] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowledge Engineering Review*, 2006.

[21] S. Singh, M. Kearns, D. Litman, and M. Walker. Reinforcement learning for spoken dialogue systems, 1999.

[22] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, 3rd edition, March 1998.

[23] J. D. Williams and S. Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech Language*, 2007.