2005－SLP－55　(16)
2005／2／5

Improving User Feedback Using Generalizations of Reinforcement Learning in Spoken Dialogue Systems

Matthias Denecke, Kohji Dohsaka

NTT Communication Sciences Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Souraku-gun, Kyoto, 619-0237, Japan,
{denecke,dohsaka}@atom.brl.ntt.co.jp

Abstract　In spoken dialogue systems, the design of efficient and user friendly dialogue strategies is an important problem. In the past, machine learning approaches using reinforcement learning have been proposed. Using reinforcement learning to optimize dialogue strategies has strong appeal as a dialogue system can be optimized using qualitative user feedback only. However, the collection of the necessary data is costly since the learning process is data-intensive. For these reasons, learning methods that make efficient use of training data are desirable. In this paper, we describe two reinforcement learning techniques that find approximative solutions, and can therefore learn more efficiently. We optimize a spoken dialogue system using both methods and compare the user feedback for the optimized system. Both methods improve the user feedback compared to the unoptimized system.

対話システムにおける強化学習の一般化手法を用いたユーザフィードバックの改善

Matthias Denecke　堂坂浩二

日本電信電話 (株), NTT コミュニケーション科学基礎研究所
〒 619-0237 京都府相楽郡精華町光台 2-4
{denecke,dohsaka}@atom.brl.ntt.co.jp

あらまし　対話システムにおいて, 効率的でユーザに使いやすい対話戦略のデザインは重要な問題である. 従来より強化学習を用いた機械学習によるアプローチが提案されている. 強化学習を用いた対話戦略の最適化は, ユーザからの定性的なフィードバックだけを用いて対話システムの最適化が行えるという特徴を持つ. しかし, 学習のために大量のデータが必要なため, データ収集のコストが大きい. このため, 訓練データを効率的に用いるような学習方法が望ましい. 本稿では, 近似解を見つけ, 効率的に学習を行うことのできるような 2 つの強化学習の手法を提案する. この 2 つの手法を用いて対話システムの最適化を行い, そのシステムへのユーザのフィードバックを比較した. どちらの手法も最適化を行わなかったシステムに比べてユーザのフィードバックを改善することができた.

1. Introduction

The responsibility of a dialogue manager includes to select appropriate actions that lead the users to their intended communicative goals. This is not a trivial task due to the presence of speech recognition errors and out-of-domain utterances from the users. For this reason, learning to choose ”right” actions from past experience is desirable. Learning in spoken dialogue systems is often formulated as the optimization of a Markov decision process in which the reward is given by positive or negative user experience. Reinforcement learning is used to carry out the optimization.

The formulation of dialogue management as a Markov decision process has been proposed by Levin and Pieraccini [1]. The optimization of the decision process using reinforcement learning has been previously investigated in Walker et al [2] and Singh et al [3], among others. In these approaches, feedback from users of an initial system is used to improve the dialogue policy. In order to come up with a working system for data collection, the initial dialogue policy has been handcrafted, leaving the ”difficult” decisions to be discovered to the learning algorithm. While users interact with the initial system, the policy state space is explored. Due to the initial hand crafted strategy, the actions of the initial system are sensible, yet not necessarily optimal. At the end of the dialogue, the users provide feedback of -1, 0 or 1, depending on the quality of the dialogue. After data have been collected from the users, the transition probabilities are learned by applying a standard value iteration algorithm.

A problem in this approach is that the size of the search space is large compared to the amount of data that can realistically be collected from users. To address this problem, Goddeau and Pineau [4] use many-to-one mappings from states and actions to backup states and actions in order to reduce the size of the search space.

Williams and Young [5] describe a method to bootstrap this initial dialogue strategy from a small corpus using supervised learning methods. Roy et al [6] model the dialogue using a Partially Observable Markov Decision Process.

Due to the scarcity of the training data, approximate solutions, as are standard in the reinforcement learning community, have a strong appeal. In this paper, we describe two methods to learn dialogue policies that reduce the size of the original problem.

2. Reinforcement Learning for Dialogue Policies

2.1. Markov Decision Processes

A Markov decision process (MDP for short) is defined by a tuple $\langle S, A, P, R\rangle$ where $S = \{s_1, \dots, s_n\}$ is a finite set of states, $A = \{a_1, \dots, a_m\}$ is a finite set of actions , $P(s'|s,a)$ is the transition model representing the probability of making a transition to state s' when taking action a in state s, and $R(s,a,s')$ is the reward for taking the transition from state s to state s' by taking action a. We define the expected reward as

$$R(s,a) = \sum_{s' \in S} P(s'|s,a) R(s,a,s')$$

Reinforcement learning is the problem faced by an agent that must learn policies through trial and error interaction with its environment. The agent bases its decision at time t on an estimation of the *action value function* $Q_t(s,a)$ (value function for short) which estimates ”how good” it is to select action a in state s. Information on the success (or absence thereof) of the actions taken is used to change $Q_t(s,a)$. This is done in such a way that the value function of successful state-action combinations converges to higher values than the value function of unsuccessful ones.

Every MDP has a optimal policy π^* which maximizes the expected discounted return of every state. There are several ways to discover the optimal policy. The exact Q values for all state-action pairs can be found by solving the linear system of Bellman equations:

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) Q(s', \pi(s')) \qquad (1)$$

The Bellman equations can be solved by *value iteration* (Kaelbling et al. [7]). Value iteration repeatedly updates the value function according to

$$Q_{t+1}(s,a) \leftarrow R(s,a) + \lambda \sum_{s'} P(s' \mid s,a) \max_{a'} Q_t(s',a') \qquad (2)$$

where $P(s' \mid s,a)$ and $R(s,a)$ are the estimated transition and reward models, respectively. It can be shown that the repeated application of equation (2) converges towards a value function Q^* and that the policy

$$\pi^*(s) = \text{arg max }_a Q^*(s,a) \qquad (3)$$

is optimal within a small error bound.

Alternatively to value iteration, the optimal solution can be found by *policy iteration* ([8]). This is done in two alternating steps. In the first step, *value determination*, the value function is determined for a given policy π^t according to equation (1). *Policy improvement* defines the next policy as

$$\pi^{t+1}(s) = \text{arg max }_a Q^t(s,a) \qquad (4)$$

The problem of learning dialogue policies from observed interactions using randomly selected actions is characterized as an *off-line, off-policy* learning problem. It is off-line as the learning takes place after the interaction completes. It is off-policy as the exploration of the state-action space does not follow a policy to be improved.

3. Our Approach

3.1. Problem

Reinforcement learning works because feedback of the user at the end of the dialogue is re-distributed over the actions taken during the dialogue. It turns out that for many applications the state and action space is too large relative to the amount of available feedback for the action value function to converge toward something useful.

In many tasks, most states encountered will not have been experienced during training. In others, as the search space increases, more data is needed to learn appropriate Q value functions as the Bellman update uses a weighted average to determine the value backup. This can be illustrated by considering the value function Q.

We recall that the result of the training process is an estimate of the value of applying action a in state s, as represented by the value function $Q(s,a)$. Dialogue states not encountered during training have their value function equal to 0 for all actions a. In other words, the training does not provide any information as to what action to apply in previousy unencountered states. Dialogue states encountered rarely during training have an estimate of the value function that is brittle. For example, if some actions have not been seen during training in certain states, the value function for these actions is 0. In other words, the training data does not provide information whether the unseeen actions in these states have a higher value than those that have been seen.

In Singh et al [3], the problem of data sparseness was addressed in carefully selecting features that appropriately represent each dialogue state. As far as reinforcement learning is concerned, each dialogue state is represented by a set of characteristics relevant for dialogue management. Several dialogue states can potentially map to the same feature vector. Therefore, the updates in the Markov decision process (in which the feature vectors identify sets of dialogue states) can be considered as non-local updates with respect to the original dialogue states. The kind of non-locality is determined by the selection of features and requires expert knowledge. However, even here, the dialogue state space becomes too large for more complex systems.

The question arises if the information in the value functions can somehow be condensed to learn more efficiently. For these reasons, there have been investigations on how to generalize the value function based on a limited subset of states experienced during exploration. This approach is referred to as *function approximation*, since the value function of unknown state action pairs is approximated with the help of previously encountered ones.

However, problems may arise due to the way in which the estimation of the value function and reinforcement learning interact. This is due to the fact that an update of the value function is not local to (s, a), but typically affects the estimated value of other state action pairs as well. Therefore, there are cases in which function approximation is unstable.

3.2. Overview of proposed Solution

We collected training data using a dialogue system in which actions were selected randomly. Analysis of the logs of these interactions reveals that relatively few dialogue states are visited relatively often. In total, 185 states were visited. The twelve most often visited states account for about half of the total number of state visits. These data suggest that an approximation of the optimal solution may be sufficient provided that the visiting frequency is taken into account during approximation.

In this paper, we investigate two approaches of approximation. Both approaches take the frequency of state visits into account and tend learn actions in frequently visited states better. The first approach consists of applying Stable Function Approximation proposed by Gordon [9] to spoken dialogue processing. Gordon proposes to approximate the action value function at given states by the weighted average of exact (i.e. not approximated) action value functions at other states. He proceeds to show that under certain conditions, stable convergence is guaranteed. We adapt his method by approximating the value function for infrequently visited states by the weighted average of similar states. This requires a measure of similarity between states which we will introduce below.

Our second approach, referred to as *State Aggregation*, consists of forming state clusters of dialogue states and consider the clusters states in an approximated Markov decision process. The key to achieving good results is to form clusters of those states that are ”similar” to each other. Two states are similar if their representations are similar and the optimal action taken in each state are the same. This leads to a chicken-and-egg problem since in order to determine the optimal action to be taken in the states, we need to form state clusters which in turn depend on the optimal action in the states. To overcome this problem, we estimate the initial Q value function using standard value iteration, and form clusters using the initial estimation. After the approximated process is solved, the Q values are projected back from the approximation to the original process. This is repeated until the optimal actions in the original process do not change much.

Due to limitations of space, we do not address the problem of adequate representations of dialogue states and actions in this paper. A detailed description of the used representations can be found in [10].

4. Stable Function Approximation

4.1. Overview

Gordon [9] proves stable convergence of value iteration with function approximation for a large class of approximators he calls *averagers*. The idea of his work is that the action value function at a given point can be approximated by the weighted average of exact (i.e. not approximated) action value functions at other points. Under certain conditions, convergence is guaranteed.

4.2. Application of Stable Function Approximation to Dialogue Processing

In what follows, s, s', s'' and t represent states of the Markov decision process. Furthermore, a, a' and a'' represent actions of the dialogue manager. Following Gordon [9], the application of the standard Bellman value update operator is replaced with a two-step update. First, the standard Bellman update (see equation 2) is applied for those states s that are frequently visited:

$$Q'(s,a) = R(s,a) + \lambda \sum_{s'} P(s' \mid s,a) \max_{a'} Q(s',a') \quad (5)$$

Subsequently, an averaging operator determines the Q values for the remaining state action pairs (t, a):

$$Q''(t,a) = \sum_{s'} \sum_{a'} \beta_{tas'a'} Q'(s',a')$$

If t is an exact state we set $\beta_{tas'a'} = 1$ for $t = s$ and $a = a'$, and 0 otherwise. If t is an approximated state, we set $\beta_{tas'a'} = 0$ for $a \neq a'$. For the remaining β, we set $\beta_{tas'a} = N/d(t, s')$, where N is a normalization factor so that $\sum_{s'} \beta_{tas'a} = 1$. Substituting $Q'(t, a)$ for equation 5, we obtain for approximated states t:

$$\begin{aligned} Q''(t,a) &= \textstyle\sum_{s'} \sum_{a'} \beta_{tas'a'}(R(s',a')+ \\ &\quad \lambda \textstyle\sum_{s''} P(s'' \mid s',a') \max_{a''} Q(s'',a'')) \\ &= \textstyle\sum_{s'} N/d(t,s')(R(s',a)+ \\ &\quad \lambda \textstyle\sum_{s''} P(s'' \mid s',a) \max_{a''} Q(s'',a'')) \end{aligned}$$

where s' ranges over exact states only. A value backup during value iteration becomes then $Q(t, a) \leftarrow Q''(t, a)$.

The optimal action a^* to be applied in state s is determined according to

$$a^* = \text{arg max }_{a \in \mathcal{A}(s)} Q_t(s, a) \tag{6}$$

5. State Aggregation

5.1. Overview

The principal idea behind our second approach, referred to as *state aggregation*, is to aggregate ”similar” states in state clusters so as to obtain a smaller decision process. We form state clusters by requiring (i) that a certain number of feature functions for all states in the same cluster yield the same value and (ii) that the optimal action arg max $_a Q(s, a)$ yield the same action a^* with a high probability when a state in the cluster is visited.

Using these concepts, our approach can be summarized as follows. We begin by estimating the transition and reward model from the observed dialogue logs. We then perform standard value iteration to obtain an estimate of the value function Q. We use this value function to form clusters of dialogue states. Using the clusters, we obtain an aggregated Markov Decision Process in which each cluster forms a new state. We solve the aggregated process using value iteration. After the process is solved, we de-aggregate to obtain an estimate of the value function for the original process. If necessary, we repeat the process.

In order to implement this approach, we need to address two issues. First, we need to determine how the transition and reward models are mapped from the original decision process to the reduced decision process and *vice versa*. Second, we need to determine how the clusters are formed.

5.2. Construction of the reduced Decision Process

In order to construct the reduced decision process, we introduce an aggregation function $c : \{1, \dots, n\} \to \{1, \dots, k\}, k < n$, assigning to each state one out of k clusters $S_1, \dots, S_k$, ignoring for the moment how this function is determined. We introduce the sampling probability $q(s|S)$ to be the probability that state s_i is chosen if the system is in cluster S_j. The probability $q(s|S)$ can be calculated according to

$$q(s_i|S_j) = \frac{P(s_i)}{\sum_{s \in S_j} P(s)} \text{ for } s_i \in S_j \tag{7}$$

where $P(s)$ is the probability that the system is in state s. We determine transition probabilities for the aggregated states in terms of the sampling probability and the transition probabilities of the original Markov decision process as follows

$$P(S'|S, a) = \sum_{s \in S} \sum_{s' \in S'} q(s|S) P(s'|s, a) \tag{8}$$

Likewise, we calculate the reward function for the aggregated states in terms of the sampling probability and the reward function of the original Markov process as in

$$R(S, a, S') = \sum_{s \in S} \sum_{s' \in S'} q(s|S) P(s'|s, a) R(s, a, s') \tag{9}$$

Equations (8) and (9) can be seen as approximations of the transition probabilities, with the sampling frequency (7) serving as the weight. Using these equations, we can construct the approximated decision process. We use standard policy evaluation/policy improvement [8] to solve the approximated process. After the approximated process is solved, we deaggregate it to obtain an approximation of the value function of the original process as follows:

$$Q(s, a) = R(S, a) + \gamma \sum_{s'} P(s'|s, a) Q(s', \pi(s')) \tag{10}$$

We iterate the process until the aggregation process becomes stable. Since the previous iteration, the value function may have been changed. Therefore, we need to update the value function. We do this according to

$$Q(S, a) = \frac{\sum_{s \in S} P(s) Q(s, a)}{\sum_{s \in S} P(s)} \tag{11}$$

and process with equation 8.

5.3. Aggregation functions

We now turn to the question how an appropriate aggregation function c can be found. The intuition behind state aggregation is to group states that are ”similar” with respect to the system dynamics. As outlined in the previous section, dialogue states are represented using feature functions that isolate aspects of the dialogue state relevant to dialogue management. Thus, any given dialogue state s can be uniquely represented as a set of constraints $f_1(s) = v_1, \dots, f_l(s) = v_l$. This is considered a cluster of size 1, since exactly one dialogue state fulfills the given constraints. Removing constraints from the set leads to a larger cluster. Thus, the approach we pursue is, proceeding bottom to top, to start out with a cluster of size 1 and perform a search to determine as

State	f_1	f_2	Times visited	Optimal Action
1	0	0	10	1
2	0	1	1	2
3	1	0	5	1
4	1	1	1	3

Table 1: Clustering example

many constraints as possible to be removed while ensuring that certain integrity constraints remain valid for the cluster.

We are interested in clustering dialogue states that ”behave similarly”. Assume for a moment that the value function Q is known. Then, we repeatedly determine the smallest sets of constraints such that arg max $_a$ $Q(s, a) = a^*$ with probability p, for some a^* and for all s that fulfill the remaining constraints. To illustrate the approach, consider table 1, showing an example of four states and 2 feature functions. For $p = 0.9$, states 1 and 2 can be clustered since the probability that the chosen action is 1 is 10 / 11 $>$ 0.9. For $p = 0.8$, states 1 to 4 form a cluster, since action 1 is chosen 15 out of 17 times, which is larger than 0.8, but not larger than 0.9.

This process yields sets of constraints that determine the clusters. Note that if $p = 1$, the algorithm degenerates to the standard Q learning algorithm, and the same solution is obtained. Since we do not know Q *a priori*, we use the standard Q policy iteration algorithm to obtain an initial estimate for Q.

One important aspect of the feature functions is that they introduce the bias according to which generalization takes place. This is because clusters are formed by imposing constraints on the features of the states in that cluster. The discussion in the previous sections leads to the algorithm shown in figure 1.

```
     do
1        Learn $Q_t(s, a)$ using value iteration
2        Determine cluster function depending on $p$
3        Determine transition and reward models of
         approximated process according to equs. (8) and (9)
4        Determine $Q(S, a)$ of the approximated process
         using value iteration
5        De-aggregate the approximated process using
         equ. (10) and assign to $Q_{t+1}(s, a)$
     while $\sum_{s,a} \|Q_t(s, a)$ - $Q_{t+1}(s, a)\| > \delta$
```

Figure 1: Algorithm

6. Evaluation

In order to evaluate the proposed methods, we implemented a spoken dialogue system. Using this system, we collected dialogue logs using the random exploration strategy as well as user feedback. We then applied the described optimization methods to the dialogue system and collected additional dialogue logs as well as user feedback.

6.1. Dialogue System

We implemented a Japanese bus information system. The implemented system has 972 different dialogue states and 5 different actions.

6.2. Data Collection

Using the Japanese bus information system, we collected 500 dialogues from 50 different users. Each user had to obtain information for 10 bus trips. If the interaction with the system exceeded a certain time limit, the dialogue was stopped and the dialogue received the evaluation -1.

Of the 972 dialogue states, one third (or 324) were final states, implying that their value function is 0 for all actions. Out of those 972 states, 185 were visited during exploration. Of those, 43 were final states. Among the remaining 142 states, there were 49 states in which, when visited, always the same action was applied. The number of states in which two, three, four or five different actions were applied, is 31, 33, 19 and 10, respectively. Thus, the size of the state-action space spanned by the exploration equals $2^{31}+3^{33}+4^{19}+5^{10}$.

6.3. User Feedback

After training, we collected 250 dialogues each with the optimized systems, having users perform the same tasks under the same conditions. None of the users of the optimized dialogue system participated in the data collection.

The users in the user study had to complete a questionnaire after each dialogue. The questionnaire was designed along the lines of the Paradise evaluation framework. The questions are shown in table 2. The answers to the first question, ”Overall, was the interaction successful?”, were assigned values -2 (no), 0 (average) and 2 (yes). The answers to all other questions were assigned integer values ranging from -2 (disagree) to 2 (agree). Except for questions 4, 5 and 6, larger values are better. The average values of the given answers are shown for the exploratory system (Expl), the system optimized with stable function approximation (SFA) and the system optimized with state aggregation (SA).

Since the answers to the first question were used as the optimization criterion, we show the answer distribution in more detail in table 3.

6.4. Discussion of the results

Both approximation algorithms are capable of optimizing the underlying dialogue process. In all of the cases, users respond more favorably to the optimized system.

	Question	**Expl**		**SFA**		**SA**	
		Avg.	Std.	Avg.	Std.	Avg.	Std.
1	Overall, the quality of the dialogue was:	0.664	1.419	1.024	1.295	0.912	1.268
2	It was easy to obtain the needed information	0.696	1.455	1.342	0.996	1.216	1.119
3	I could understand what I was supposed to say	1.04	1.195	1.408	0.910	1.584	0.798
4	I had to speak unnaturally in order to make the system understand me	-0.548	1.258	-0.612	1.340	-1.004	1.180
5	I was nervous while interacting with the system	-0.784	1.223	-1.112	1.197	-1.192	1.066
6	The system irritated me	-0.924	1.228	-1.204	1.236	-1.18	1.085
7	I would like to use the system again	0.548	1.335	0.916	1.167	1.008	0.961

Table 2: User feedback. Shown are the averaged answers (avg) and their standard deviation (std) for the exploratory system (Expl), the system optimized with stable function approximation (SFA) and the system optimized with state aggregation (SA).

Answer	**Average**		
	Expl	**SFA**	**SA**
Good	47.2	59.6	53.2
Average	38.8	32	39.2
Bad	14	8.4	7.6

Table 3: Detailed user feedback for the overal quality of the dialogue. Shown are the answer percentage for the exploratory system (Expl), the system optimized with stable function approximation (SFA) and the system optimized with state aggregation (SA).

7. Summary

We described two novel methods for fast reinforcement learning of dialogue policies. We showed that the first proposed method effectively shortens the lengths of frequently visited dialogues. This is despite the fact that the state-action space is considerably larger than in previously reported work. However, the optimized dialogue strategy does not do much for infrequently visited states as these are excluded from training.

8. References

[1] E. Levin and R. Pieraccini, "A Stochastic Model of Human Computer Interaction for Learning Dialog Strategies," in *Proceedings of Eurospeech, Rhodos, Greece*, 1997.

[2] M. Walker, J. Fromer, and S. Narayanan, "Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email," in *Proceedings of ACL/COLING 98*, 1998.

[3] 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.

[4] D. Goddeau and J. Pineau, "Fast Reinforcement Learning of Dialog Strategies," in *IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP), Istanbul, Turkey*, 2000.

[5] J. D. Williams and S. Young, "Using Wizard-of-Oz Simulations to Bootstrap Reinforcement Learning Based Dialog Management Systems," in *Proceedings of the 4th* SIGDIAL *Workshop on Discourse and Dialogue*, 2003.

[6] N. Roy, J. Pineau, and S. Thrun, "Spoken Dialog Management for Robots," in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 2000.

[7] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[8] R. Sutton and A. Barto, *Reinforcement Learning*. MIT Press, 1998.

[9] G. J. Gordon, "Stable function approximation in dynamic programming," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[10] M. Denecke, K. Dohsaka, and M. Nakano, "Fast reinforcement learning of dialogue policies using stable function approximation," in *Lecture Notes in Artificial Intelligence*. Springer, 2005.