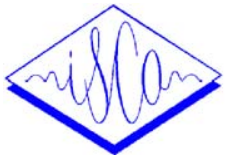
ISCA Archive
http://www.isca-speech.org/archive



INTERSPEECH 2004 - ICSLP
8th International Conference on Spoken Language Processing
ICC Jeju, Jeju Island, Korea
October 4-8, 2004

Learning Dialogue Policies using State Aggregation in Reinforcement Learning

Matthias Denecke, Kohji Dohsaka, Mikio Nakano

Communication Sciences Laboratories, NTT
NTT Corporation
Morinosato Wakamiya 3-1
Atsugi, Kanagawa 243-0198, Japan
{denecke,dohsaka,nakano}@atom.brl.ntt.co.jp

10.21437/Interspeech.2004-149

Abstract

The learning of dialogue strategies in spoken dialogue systems using reinforcement learning is a promising approach to acquire robust dialogue strategies. However, the trade-off between available dialogue data and information in the dialogue state either forces information to be excluded from the state representations or requires large amount of training data. In this paper, we propose to use dynamic state aggregation to efficiently learn dialogue policies using less data. State aggregation reduces the size of the problem to be solved. Experimental results show that the proposed method converges faster and that in case of data sparseness, the proposed method is less sensitive to atypical training examples.

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 application of reinforcement learning to spoken dialogue systems has been previously investigated in Walker et al [1] and Singh et al [2]. The authors use the feedback from users of an initial system to improve the dialogue policy. In order to come up with a working system for data collection, the initial dialogue policy has been hand-crafted, 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. For this reason, approximate solutions, as are standard in the reinforcement learning community, have a strong appeal. In this paper, we investigate state aggregation as a technique to approximate the Markov decision process. We show how we can obtain acceptable dialogue policies using less training data by applying a form of averaging and generalization during reinforcement learning.

In this paper, we propose a method to learn dialogue policies that reduces the size of the problem by aggregating dialogue states to state clusters. Instead of optimizing dialogue policies for each state, we optimize dialogue policies for each cluster, thus effectively reducing the problem size. Experimental results show that the solutions of the proposed method are closer to the optimal solution. Furthermore, since the formation of clusters can be seen as an averager, the proposed method is less sensitive to the presentation of atypical training examples.

2. Reinforcement Learning for Dialogue Policies

2.1. Markov Decision Processes

A Markov decision process is defined by a tuple $\langle S, A, P, R\rangle$ where $S = \{s_1, \ldots, s_n\}$ is a finite set of states, $A = \{a_1, \ldots, a_m\}$ 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

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

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 increase or decrease $Q_t(s,a)$. 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. The exact Q values for all state-action pairs can be found by solving the linear system of Bellman equations:

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

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; in this paper, we focus on policy iteration which consists of iteration through a sequence of monotonically increasing policies. 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

$$\pi^{t+1}(s) = \arg\max_a Q^t(s,a) \tag{2}$$

2.2. Data Sparseness Issues

Reinforcement learning works because feedback of the user at the end of the dialogue is re-distributed over the actions taken during the dialogue. However, 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. The question then arises if the information in the value functions can somewhat be condensed to learn more effectively.

Moreover, due to data sparseness, adding a new dialogue to the training set can result in a substantially different dialogue strategy if the new dialogue is somewhat atypical. This is due to the fact that the new dialogue will modify the value functions in such a way that a different action is selected (see equation (2)).

3. Our Approach

The principal idea behind our approach is to aggregate "similar" states in state clusters in such a way that we obtain a smaller decision process. In order to determine similarity of states, we introduce feature functions that indicate features of states that are relevant for dialogue management. For example, the number of filled or confirmed slots would be such a feature.

We form clusters by requiring (i) a certain number of feature functions of states in the same cluster to yield the same value and (ii) the value function $Q(s,a)$ to yield the same action a^* with a high probability.

Using these concepts, our approach can then be summarized as follows. We 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.

3.1. State Aggregation

The idea behind state aggregation is to group "similar" states into clusters and perform value iteration for the newly created clusters. In order to do that, we introduce a function $c : \{1,\ldots,n\} \rightarrow \{1,\ldots,k\}, k < n$, assigning one out of k clusters $S_1,\ldots,S_k$ to each state. 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 which can be calculated according to

$$q(s_i|S_j) = \frac{P(s_i)}{\sum_{s\in S_j} P(s)} \text{ for } s_i \in S_j \tag{3}$$

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

$$P(S'|S,a) = \sum_{s\in S}\sum_{s'\in S'} q(s|S)P(s'|s,a) \tag{4}$$

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

$$R(S,a,S') = \sum_{s\in S}\sum_{s'\in S'} q(s|S)P(s'|s,a)R(s,a,s') \tag{5}$$

Equations (4) and (5) can be seen as approximations of the transition probabilities with the sampling frequency (3) serving as the weight.

We now have an approximation to the original Markov process. We use standard policy evaluation/policy improvement [3] to solve the approximated process. After the approximated process is solved, we de-aggregate it to obtain an approximation of the value function of the original process as follows:

$$Q(s,a) = R(S,a) + \gamma \sum_{s'} P(s'|s,a)Q(s',\pi(s')) \tag{6}$$

We iterate the process until the aggregation process becomes stable. Since before then, the value function has changed, we need to update the value function. We do this according to

$$Q(S,a) = \frac{\sum_{s\in S} P(s)Q(s,a)}{\sum_{s\in S} P(s)} \tag{7}$$

and process with equation 4.

The last remaining question is the choice of an appropriate cluster function c. The intuition behind state aggregation is to group states that are "similar" with respect to the system dynamics. We defer this discussion to section 3.3, after describing the representations, as the "similarity" of states is dependent on the representations.

3.2. Representations

3.2.1. Representation of the State space

In our approach, following [4], we partition the state space according to the information they contain with respect to action selection. We do this by introducing *feature functions* $f_i(s)$ that determine aspects of the dialogue state which are relevant for dialogue management.

The complete abstract dialogue state space is shown in table 1. In the following, we refer to the set of feature functions as V. We note that not all combinations of variable assignments actually represent sensical dialogue states as not all variables are independent of each other.

Feature function	Possible Values	Size
Confirmation	*all unconfirmed, some unconfirmed, all confirmed*	3
Length of input	*short, intermediate, long, very long*	4
Information	*no filled, some filled, all filled*	3
Minimal Confidence	*low, medium, high*	3
Maximal Confidence	*low, medium, high*	3
Intention	*selected, determined, finalized*	3
Total size		972

Table 1: Abstract dialogue state space.

We assume that at any time in the dialogue, there is a set of slots to be filled. The set of slots is the union of the slots specified in the active dialogue goals. A slot i is called $needed$ if there is a goal currently active (i.e. not deselected) that requires slot i to be filled. Thus, the number of needed slots decreases as more and more goals become deselected. For convenience, we define N_t and F_t the set of needed slots, and filled slots, at time t, respectively. For each slot i, we have feature functions $filled_t(i)$ and $confirmed_t(i)$ evaluating to $true$ if the ith slot has been filled, or filled and confirmed, respectively. Furthermore, there is a function $prompted_t(i)$ returning the number of times the ith slot has been prompted. Finally, there is a function $confidence_t(i)$ returning $low, medium$ or $high$, depending on the confidence with which the value in the slot has been recognized. If $filled_t(i)$ is $false$, $confirmed_t(i)$ is undefined.

The variable *Confidence* evaluates to the minimal confidence score of all filled slots, or

$$Confidence_t = \min_{i \in N_t(i) \cap F_t(i)} confidence_t(i) \quad (8)$$

and is set to high at the beginning of the dialogue. Variable *Information* evaluates to $allFilled$, $existsFilled$ or $allUnfilled$ according to

$$Information_t = \begin{cases} allFilled & \forall i : filled_t(i) \\ allUnfilled & \forall i : \neg filled_t(i) \\ existsFilled & otherwise \end{cases} \quad (9)$$

The variable *Confirmation* is evaluated to $allUnconfirmed$, $existsUnconfirmed$ and $allConfirmed$ along the lines of the evaluation of variable $Information$.

The feature function *Intention* represents the degree to which the intention of the user could be determined. The value *selected* indicates that the service the user would like to invoke has not yet been determined. The value *determined* signifies that the intended service could be determined but there are some unknown or unconfirmed parameters. Finally, the value *finalized* indicates that the both the intended service as well as all relevant parameters are established in the discourse. It follows that all dialogue states for which the function *Intention* evaluates to *finalized* are final dialogue states.

3.2.2. Representations of Actions

We chose to support the five action classes shown in Table 2. *Explicit confirmation* refers to an action in which the filler of one slot is explicitly confirmed. *Prompt one* and *Prompt multiple* refer to actions in which one or more fillers are prompted. *Implicit Confirm, Prompt one* and *Implicit Confirm, Prompt multiple* are as above, except that slot fillers are implicitly confirmed. It should be noted that the decision process does not have any information on the slots a selected action is to be applied. It is the responsibility of the dialogue manager to determine, for example, which slot to confirm, if the action $explicitconfirmone$ is selected.

Name	Confirmation	Information	Multiple slots
Expl. confirmation	y	n	n
Prompt one	n	y	n
Prompt multiple	n	y	y
Impl. conf., prompt one	y	y	n
Impl. conf., prompt multiple	y	y	y

Table 2: Action classes supported in the system.

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 3: Clustering example

3.3. Cluster functions

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 $f_1(s) = v_1, \ldots, f_l(s) = v_l$. 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.

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 $\text{argmax}_a\ Q(s,a) = a^*$ with probability p, for some a^* and for all s that fulfill the cluster constraints. 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 w do not know Q *a priori*, we use the standard Q policy iteration algorithm to obtain an initial estimate for Q. Table 3 shows an example for 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.

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.

4. Evaluation

We implemented a Japanese bus information system. We encoded the abstract state and action spaces as described in the

States	# of states
Intermediate	142
1 action	49
2 actions	31
3 actions	33
4 actions	19
5 actions	10
Final	43

Table 4: Statistics on visited dialogue states

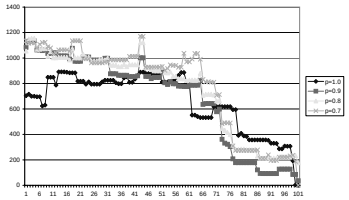


Figure 1: Distance to optimal solution of standard Q learning results and state aggregation results.

previous sections.

4.1. Data Collection

For training purposes, we collected a corpus of 500 dialogues from 50 different users. Each user had to fulfill a task consisting of 10 dialogues. The task was presented to the user on paper. After presentation, the user interacted with the system by voice only to acquire the information necessary to complete the task. At the end of the dialogue, user indicate whether the experienced dialogue was good, average or bad. Actions of the dialogue system were chosen randomly among all actions that were applicable at that point in the dialogue.

The implemented system has 972 different abstract dialogue states and 5 different abstract actions. Of the 972, one third (or 324) were final states, implying that their value function is 0 for all actions. Theoretically, the search space could be $5^{972*\frac{2}{3}}$, however, the pre- and postconditions on the actions limit the search space further. In addition, not all combinations of values of the state variables lead to valid abstract states which limits the number of observable states. The following considerations give some indication on the size of the search space.

Out of the 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. In the remaining 93 states, at least two different actions were applied during the data collection. Table 4 gives information on the distribution of actions among the states. The size of the state-action space spanned by the exploration equals $2^{31}+3^{33}+4^{19}+5^{10}$, however, the state-action space that could potentially be explored is much larger.

4.2. Distance to Optimal Solution

To evaluate our method, we determine the exact solution of the Markov Decision Process using Q learning. Following that, we determine the number of states in which the actions differ from the exact solution if less dialogues are presented. The result is shown in figure 1. We display the graphs for $p = 1$ (exact solution), $p = 0.9$, $p = 0.8$ and $p = 0.7$. It can be shown that in the presence of data sparseness, the solutions generated by the proposed method are closer to the exact solution that standard Q learning. Moreover, the approximated solutions are less susceptible to be influenced by the presentation of unusual dialogues. More specifically, in standard Q learning, adding one dialogue to the training set caused the switch the optimal actions in up to 19 out of 93 states. Using the approximative solution for $p = 0.8$, this number was reduced to 7.

5. Conclusions

We proposed a method to learn dialogue policies based on user feedback using reinforcement learning. The problem in this approach is the data sparseness of distributing user feedback to the dialogue states in which actions have to be taken. It could be shown that state aggregation is an effective means to determine better approximations of the exact solution. In [5], we propose an alternative method of approximation that takes into account the frequency with which dialogue states are visited.

The proposed state representation can be seen as a static state aggregation as several concrete dialogue states for which the feature functions evaluate to the same values are treated equivalently. An interesting question that remains to be investigated is where to draw the line between the dynamically and statically aggregated states.

6. Acknowledgements

We would like to thank Shoji Makino and all members of the Dialogue Understanding Research Group for support and helpful discussions.

7. References

[1] 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.

[2] 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.

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

[4] M. Denecke, "Informational Characterization of Dialogue States," in *Proceedings of the 6th International Conference on Speech and Language Processing, Beijing, China*, 2000.