Efficient Model Learning for Dialog Management

Finale Doshi and Nicholas Roy
MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar Street
Cambridge MA, 02139 USA

Introduction

Spoken dialog managers can allow for natural human-robot interaction, but noisy voice recognition and linguistic ambiguities can make it difficult to decipher the user's intent. Planning algorithms such as Partially Observable Markov Decision Processes (POMDPs) have succeeded in dialog management applications because they are robust to uncertainties in the dialog. Like all dialog planning systems, however, POMDPs require an accurate model of the user's communication style and preferences. POMDPs are generally specified using large probabilistic models with many parameters. These parameters are difficult to specify from domain knowledge, and gathering enough data to estimate the parameters accurately *a priori* is expensive.

We use a Bayesian approach to learning the user model. First, we show how to compute the optimal dialog policy with uncertain parameters (in the absence of learning). Second, we present a heuristic that allows the dialog manager to intelligently replan its policy given data from recent interactions. Our approach performs well both in simulation and on a dialog manager of a robotic wheelchair (Figure 1). Finally, we present an alternative to our first approach which has potential for learning user preferences more robustly.

Figure 1: Our dialog manager allows more natural human communication with a robotic wheelchair.

Related Work

Most dialog managers assume that an accurate user model is available ((Williams, Poupart, & Young 2005), (Pineau, Roy, & Thrun 2001), (Williams & Young 2005)). In the field of POMDP learning, the Medusa algorithm (Jaulmes, Pineau, & Precup 2005) places priors over the POMDP parameters, samples and solves many POMDPs from these prior distributions, and uses this sample to vote for a good policy. While principled, Medusa is computationally expensive. Other work uses minimax approaches to find robust MDP policies when transition matrices are uncertain (Nilim & Ghaoui 2004). We follow Medusa's Bayesian approach; however, we begin with only one POMDP dialog manager instead of several samples.

Maximizing Expected Performance

Acting with Uncertain User Models

A POMDP-tuple contains the sets S (states), A (actions), and O (observations). In our case, the states represent the user's (hidden) intent, while the observations represent the actual utterances the dialog manager receives from the voice recognition software. The functions $R(s,a)$ (reward for taking action a in state s), $T(s'|s,a)$ (state transition probability $P(s'|s,a)$), and $\Omega(o|s,a)$ (observation probability $P(o|s,a)$) together model the user's communication style and preferences. Since we are uncertain about the user model, we place priors over the parameters in R,T, and Ω. We assume the true parameter values do not change. Figure 2 shows a simple dialog POMDP.

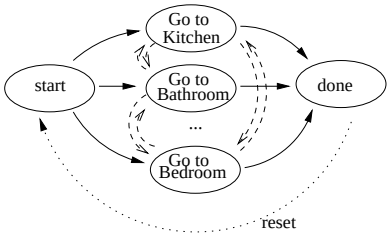

Figure 2: Our dialog POMDP. Solid lines represent more likely transitions; we assume that user is unlikely to change their intent before their original request is fulfilled (dashed lines). The system resets once we enter the 'done' state.

An optimal POMDP policy maps beliefs b (distributions over states) to actions to maximize the total expected reward. For example, if the user is likely in state s_1, but may be in s_2, the optimal action may be to query the user if they are truly in s_1. The optimal value function $V(b)$ represents the total reward we expect to gather if the dialog manager starts with an initial belief b and acts optimally. $V(b)$ can be found via the following (standard) recursion:

$$V_n(b) = \max_{a \in A} Q_n(b, a), \quad (1)$$

$$Q(b, a) = \max_i \vec{q_a} \cdot b, \quad (2)$$

$$q_a(s) = R(s, a) + \gamma \sum_{o \in O} \sum_{s' \in S} T(s'|s, a)\Omega(o|s', a)\alpha_{n-1,i}(s) \quad (3)$$

where $\Omega(o|b, a)$ is the probability of seeing o after performing a in belief b.[1] Each iteration of this recursion is called a *backup*. When the value function converges to (within ϵ of) a unique set of values, we have *backed up to ϵ-convergence*.

The $Q(b, a)$ values represent the total expected reward if we start from b, do a, and then act optimally. Equation 1 chooses the action that maximizes the expected reward to derive the optimal dialog policy. Equations 2 and 3 take expectations over the belief (uncertainty in user's intent) and the user model (ambiguities in the user's communication). If the user model contains uncertain parameters, then we must add an additional expectation in Equation 3:

$$\begin{aligned} q_a(s) &= E_M[R(s, a) + \\ &\quad \gamma \sum_{o \in O} \sum_{s' \in S} T(s'|s, a)\Omega(o|s', a)\alpha_{n-1,i}(s)] \\ &= E_M[R(s, a)] + \\ &\quad \gamma \sum_{o \in O} \sum_{s' \in S} E_M[T(s'|s, a)]E_M[\Omega(o|s', a)]\alpha_{n-1,i}(s)] \end{aligned}$$

The second line follows from the linearity of expectations and since the parameter distributions T and Ω are independent. Thus, we show solving the POMDP with the expected parameter values will maximize the expected reward.

Efficient Policy Updates after Learning

After each user interaction, we update a Gaussian reward prior and Dirichlet observation and transition priors. Updating Dirichlet and Gaussian priors is straight-forward, but we need to know the user's (hidden) state to update the correct $R(s, a)$, $T(s'|s, a)$, and $\Omega(o|s, a)$ distributions. The simple structure of the dialog model allows us to infer the user's state history from the action that ended the exchange.

Recomputing a new policy after each dialog would require large amounts of computation. Instead, we update the current dialog policy using the new expected values of the parameters. How many backups should we perform? Backing up to convergence is computationally expensive and may be a wasted effort if the parameters are fairly uncertain. We tested three heuristics: backing up to convergence, backing up once, and backing up proportionally to the variance reduction in the parameters. The intuition behind the final heuristic is that we wish to expend the more computational effort when parameters become more certain.

[1] This representation uses the fact that the value function is piecewise convex and linear. We use a point-based value iteration method to approximate the solution (Pineau, Gordon, & Thrun 2003).

Simulation Results

We first tested our approach on a simulated dialog manager. The user's states corresponded to five goal locations; actions included clarification queries (low cost) and motions (high cost if incorrect). Our priors, used to compute the initial dialog management policy, guessed observations had half the true error rate and that the user was more forgiving of mistakes. Averaged over 100 simulated trials, all heuristics improved performance similarly, but backing up proportionally to variance reduction did so with less computation.

Wheelchair Results

We also tested our algorithm on a real wheelchair dialog manager. More complex than the simulation model, the wheelchair's vocabulary included not only keywords associated with particular locations, but also several words—such as elevator—not initially mapped to any goal location. Our adaptive dialog manager learned associations for the unmapped keywords and the user's preferences (see example in Table 1). As seen in Figure 3, the learning dialog manager generally out-performed the non-learner.

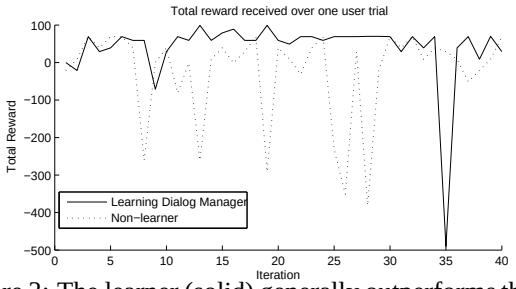

Figure 3: The learner (solid) generally outperforms the non-learner (dashed), rarely making serious mistakes.

Table 1: Part of a wheelchair-user dialog. The dialog specification failed to specify how to handle the word 'elevator.' The learner realizes that while there are multiple elevators, the user most often uses it in the context of the Gates Tower.

Without learning
USER: Take me to the elevator.
The system does not know what 'elevator' means.
ROBOT: Where did you want to go?
USER: The Gates elevator please.
ROBOT: Do you want to go to the Gates Tower?
USER: Yes.
ROBOT: Going to the Gates Tower.
After learning:
USER: Take me to the elevator.
ROBOT: Do you want to go to the Gates Tower?
USER: Yes.
ROBOT: Going to the Gates Tower.

Improving Dialog Manager Robustness

Our dialog policy maximized the expected reward over the uncertain user model, but the outliers in our user tests are unacceptable in the wheelchair domain. For more robust performance, we must consider the spread of the parameters and their mean. By including the parameters—and their uncertainty—as an explicit part of the dialog model, the dialog manager can take actions that to reduce parameter uncertainty. Such a dialog manager will also be robust to poor initial parameter estimates.

We incorporate some of the unknown parameters into the POMDP model to create a more robust dialog manager (Figure 4). Our new state space consists of a vector $\{s_u, \vec{s_r}\}$, where s_u is the user's state (where he wishes to go) and $\vec{s_r}$ is vector of the user's preferences: rewards associated with state-action pairs such as making a general query or making an incorrect confirmation. For the present, we consider learning the only user's preferences (not observation parameters) and assume a finite set of possible reward values. We extend our observation space to be $\{o_d, o_r\}$, where o_d is the speech to the dialog manager and o_r is a reward entered by the user. Our new belief b($s_u, \vec{r}$) represents the probability that the user is in state s_u *and* the rewards are given by $\vec{s_r}$.

Including the reward parameters in the POMDP increases the size of the state space and thus increases the computation required. As seen in Figure 4, however, independencies between the user's current goal (s_u) and his overall reward preferences ($\vec{s_r}$) allow us to factor transition and observation probabilities in the POMDP. For example, consider the probability of observing a particular $\{o_d, o_r\}$ in a state $\{s_u, \vec{s_r}\}$. The observed speech input does not depend on the user's reward model, so the observation probability factors as:

$$P(o_d, o_r | s_u, \vec{s_r}) = P(o_d | s_u) \cdot P(o_r | s_u, \vec{s_r})$$

Similar factorizations for other transition and observation probabilities make solving this complex POMDP tractable.

Our new POMDP dialog manager is aware of its uncertainty in the model, but the only way for it to discover the consequences of a poor decision is to make a bad choice and experience its effects. To further increase robustness, we add a set of meta-queries: "I am x% certain that you wish to go to the parking lot. In the future, should I confirm this with you or proceed directly to the goal location?" If the user responds that the dialog manager should confirm the action in the future, we can infer that the user places a high penalty on incorrect decisions *without experiencing its effects*. However, if the user tells the wheelchair to proceed without confirmation, we have learned that the user is willing to risk an incorrect decision to avoid being asked multiple questions.

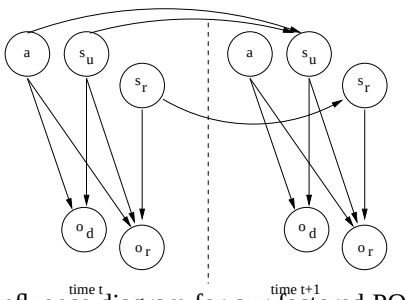

Figure 4: Influence diagram for our factored POMDP. The user state s_u depends on the current action a and the previous state, while the user's preferences s_r never change. Of the two observations, the observed reward o_r depends on both the user's state and his preferences and the observed dialog o_d depends only on the user's state.

The user's response to the meta-query allows us to prune inconsistent preference states $\vec{s_r}$ from the belief space. To determine which $\vec{s_r}$ are feasible, we first find the optimal POMDP policy without including the meta-queries. Next, for all $\vec{s_r}$ and $b(s_u)$ with x% probability mass in one state, we check if predicted action matches the user's response. Although we have to solve the dialog manager POMDP in two stages—without and with meta-queries—all computations may be performed before user interactions begin.

Conclusions

We presented two POMDP-based approaches to dialog management on a robotic wheelchair. Our first approach based its decisions only the expected values of the uncertain parameters. To improve the quality of this approach, we note that re planning only corrects for loss in performance due to incomplete convergence; by estimating the variance due to the uncertainty in both the dialog and the user model we can judge the effectiveness of additional planning.

While efficient, our first approach was not necessarily robust. Our second approach, which folds parameters into the POMDP dialog manager, promises to be robust but requires much more computation. However, we show that the appropriate factoring schemes can keep the computational load feasible. Since we are planning over the parameter space, very little computation needs to be performed online. The heuristics in this paper show that POMDP learning is promising for dialog management, and in the future we hope to create an algorithm that is efficient and learns robustly.

References

Jaulmes, R.; Pineau, J.; and Precup, D. 2005. Learning in non-stationary partially observable markov decision processes. Workshop on Non-Stationarity in Reinforcement Learning at the ECML.

Nilim, A., and Ghaoui, L. 2004. Robustness in markov decision problems with uncertain transition matrices.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for pomdps.

Pineau, J.; Roy, N.; and Thrun, S. 2001. A hierarchical approach to pomdp planning and execution. In *Workshop on Hierarchy and Memory in Reinforcement Learning (ICML)*.

Williams, J., and Young, S. 2005. Scaling up pomdps for dialogue management: The "summary pomdp" method. In *Proceedings of the IEEE ASRU Workshop*.

Williams, J. D.; Poupart, P.; and Young, S. 2005. Partially observable markov decision processes with continuous observations for dialogue management. In *Proceedings of SIGdial Workshop on Discourse and Dialogue 2005*.