A Bayesian Method for Learning POMDP Observation Parameters for Robot Interaction Management Systems

Amin Atrash
School of Computer Science
McGill University
Montreal, QC H3A 1A8
aatras@cs.mcgill.ca

Joelle Pineau
School of Computer Science
McGill University
Montreal, QC H3A 1A8
jpineau@cs.mcgill.ca

Abstract

Technology has allowed robots to enter more personal settings in our society, appearing in environments alongside humans. These new situations provide a new set of problems, including the interaction and control of the robot by untrained humans, as well as adapting to an unconstrained world designed for humans. In this paper, we address the issue of robot learning in these environments while taking advantage of a user working alongside the robot. We present a framework for gradually learning a model of the user through a parametric observation function. This type of framework allows us to begin with a rough model of the world and adjust it from experience. By relying on an oracle providing optimal policy information, we are able to learn the observation model and adjust the robot's behavior to match that of the oracle. We address the problems of learning and modifications necessary to handle the observation function and learning for rare events. We demonstrate the feasibilty of the algorithm on a robot-interaction domain and compare against a model-free method for action-selection.

INTRODUCTION

Personal robots have become increasingly ubiquitious over the last ten years. That trend is unlikely to slow down. From robotic vehicles, to intelligent wheelchairs, to social and cognitive assistants, the opportunities are immense. Designing personal robots however requires a profound paradigm shift, compared to their industrial predecessors. In particular, it is imperative that these robots be able to learn and adapt to the environment and humans that surround them. Without the ability to learn, robots are condemned to use preset models of the environment and humans, which are invariably brittle, incomplete, and often inaccurate, especially when it comes to modelling the humans in the environment. This clearly suggests there are exciting opportunities for developing learning methods that can provide personal robots with the flexibility necessary to adapt to their domain.

Consider the case of an autonomous wheelchair which must work alongside a user for an extended period of time. This scenario presents several interesting challenges. As with any robot, the wheelchair must account for noise from the sensors and environment. In particular, the wheelchair must handle input from the user through a user interface which are notoriously noisy and suffer from ambiguity due to human communication. In the case of speech input, When issuing a command, words are often transcribed incorrectly. Even when recognition is perfect, the user may not have provided enough information to determine the exact intention. For example, the user may say "move" without specifying a direction or "go to the office" without specifying a specific office. The wheelchair must be able to properly handle this type of noise and ambiguity to perform robostly and in a manner comfortable to the user.

This paper focuses on developing a system which is able to adapt to the environment while working alongside a user by taking advantage of information provided by the user. By consulting with the user, the robot is able to gain additional information which allows it to adjust the internal representation of the environment. This is a similar idea as work in imitation learning (Atkeson and Schaal 1997), as well as methods which learn model parameters from data (Rabiner 1990). To this end, we present an algorithm which relies heavily on Bayesian reinforcement learning (Dearden, Friedman, and Andre 1999; Duff 2002; Jaulmes, Pineau, and Precup 2005; Poupart et al. 2006; Ross, Chaib-draa, and Pineau 2007; Doshi, Pineau, and Roy 2008). This provides us with several advantages from the reinforcement learning paradigm, mainly the interleaving of execution and learning. We are able to use the models for execution and decision-making during the learning process. This is in contrast to some other learning algorithms which require batch data for training. Furthermore, the learning can be directed towards finding a model which s consistent with the policy, rather than simply focusing on the parameters themselves. We take advantage of the optimal policy information provided by an oracle to help infer the model parameters. This mechanism lends itself ideally to a human-robot interaction domain.

Our previous work (Atrash and Pineau 2009) presented an algorithm using these ideas for learning the reward for taking actions to adapt the robot behavior to match the intentions of the user. In this paper, we focus on extending the framework to learn the observation dynamics. We will present the framework and the modifications necessary to learn observations. Observations require a different representation and a corresponding change to the learning mechanism. We will also propose a mechanism for compensat-

ICAPS 2010 POMDP Practitioners Workshop, May 12, 2010, Toronto, Canada.

ing for rare events. This allows the models to learn the less frequent events, which are often the more important. Otherwise, the models risk learning to repeat the most common action repeatedly. We validate this learning on a wheelchair interaction manager. To highlight the importance of model learning in such domains, we show that our approach compares favorably with model-free supervised learning methods, such as SVMs. We show that the models are adapting over time and provide an advantage over the model-free methods. Finally, we discussion extensions to the learning method to allow learning of other model parameters.

BACKGROUND

This section reviews the technical material necessary to understand our approach. Our approach assumes the robot's task domain can be represented probabilistically, in the form of a POMDP model, as defined in this section. The learning component of our approach follows the Bayesian reinforcement learning paradigm, also described below.

POMDPs

Partially Observable Markov Decision Processes (POMDPs) (Kaelbling, Littman, and Cassandra 1998) are stochastic models used to model non-deterministic decision-making problems. POMDPs consist of a set of states, S, a set of actions, A, and a set of observations, Z. When an action, a, is execute in state s, the system transitions to state s' with probability $Pr(s'|s,a)$. The agent then receives a reward, $R(s,a)$ and an observation z is emitted with probability $Pr(z|s')$. The agent has an initial belief distribution across the states, $Pr(s_{t=0})$.

At any point in time, the underlying state, s, is not necessarily observable by the agent. Therefore a distribution across all states must be maintained. The belief distribution $Pr(s_t)$ is updated recursively each time the agent executes an action a and receives an observation z:

$$Pr(s_t = s') = \frac{\sum_s Pr(z|s')Pr(s'|s,a)Pr(s_{t-1}=s)}{\sum_{s''}\sum_s Pr(z|s'')Pr(s''|s,a)Pr(s_{t-1}=s)} \quad (1)$$

Given a POMDP problem, an action-selection policy π can be determined which maps belief states to actions:

$$\pi(b) \rightarrow a.$$

Solving a POMDP means finding the policy that maximizes the expected discounted return:

$$E[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)|b_0].$$

This presumes of course that the reward function (as well as the transition and observation paramters) is known.

Efficient approximate solution methods are available to solve this optimization problem, though details of these algorithms are beyond the scope of this paper. For our experiment, we use the Point-Based Value Iteration (PBVI) algorithm which approximates the policy by using stochastic trajectories to select belief points (Spaan and Vlassis 2005; Pineau, Gordon, and Thrun 2003). This method allows us to solve relatively large POMDPs in a reasonable amount of time.

The algorithms here focus on learning the observation function $Pr(z|s')$. The observation function is particularly important in human-machine interaction tasks at the observation function helps capture the error and ambiguity that occurs from the input. For a given state, s, the observation function is a multinomial. The Dirichlet distribution is the conjugate prior of the multinomial distribution. This fact will be useful to maintain a Bayesian posterior over the parameters as required by the Bayesian reinforcement learning framework.

Bayesian Reinforcement Learning

The aim of Bayesian reinforcement learning(Dearden, Friedman, and Andre 1999; Duff 2002) is to maintain a posterior distribution over possible model parameters, and to compute an action selection policy which is optimal with respect to this posterior.

A key step in all Bayesian RL methods is thus to compute the posterior over the transition and reward parameters that define domain model. This is usually done by maintaining Dirichlet distributions over possible models and updating the hyper-parameters of the Dirichlet as new events are experienced. A separate Dirichlet distribution is maintained for every (s,a) transition and observation probability distribution. It is straight-forward to update the posterior over this distribution whenever new experience is acquired if the state is known. If the new experience does not provide complete information, the update becomes more difficult.

While updating the posterior can be done easily in closed-form, it is not so easy to compute an optimal policy with respect to this posterior. Existing methods take different approaches to this problem. Some of the most recent methods (Jaulmes, Pineau, and Precup 2005; Doshi, Pineau, and Roy 2008) approximate the posterior by sampling a small set of candidate models, and solving those. The posterior over models continues to be updated, as new experience is acquired. Periodically, the set of sampled models can be re-sampled. Thus there are two mechanisms for learning: updating of the hyper-parameters, and re-sampling of models.

A Bayesian Approach for Online Learning

The Overall Approach

Our framework is presented in Algorithm 1. First, the system initializes the Dirichlet distribution based on any available prior knowledge. This can be a uniform prior if no information is available. Second, a set of models is sampled from the distributions and the optimal policy computed for each. The learning phases consists of iterations of execution and learning. An oracle is queried to obtain the optimal policy action. This actions is then executed, and the Dirichlet parameters updated. Periodically, new models are sampled and old models removed. As the Dirichlet parameters improve over time, these new models result in policies closer to the optimal policy.

Algorithm 1 Algorithm

Initialize Dirichlet Distribution
Sample n POMDPs $P_1, P_2, ...P_n$ from Dirichlet
Solve $P_1, P_2, ..., P_n$ using approximate POMDP method
loop
  Get action from oracle
  Execute Action
  Obtain Observation
  Update Belief States
  For each POMDP, M_i whose policy agreed with the oracle action
  $\alpha(s,z) \leftarrow \alpha(s,z) + \lambda f(z) Pr_M(z|s) \quad \forall s \in S, z \in Z$
  if resample_POMDP() **then**
    Remove k worst performing POMDP
    Sample k new POMDP from Dirichlets
    Find policy for new POMDPs
  end if
end loop

Policy Oracle

One of the key elements in our framework is the use of an oracle to provide an optimal policy action. This provides the learning component of our framework with information to control the update the Dirichlet parameters by only considering sample models which selected the same action. Conceptually, the models which selected the correct action are more "correct" and provide us with an estimate of the direction towards the optimal parameters

Previous work on learning using an oracle (Jaulmes, Pineau, and Precup 2005) relied on the oracle for full state information. While this is sometimes feasible, in many scenarios it is not realistic to have an oracle to provide such detailed information. Often, measurements for ground truth state information would be expensive and tedious. In the case of human-robot interaction, the underlying state may not be attainable or represents an abstract concept. Instead, we believe it is much more reasonable to request the correct action to take at the time, basically what the user would have done in that situation. This type of information is more natural and intuitive for a human to provide. For example, for robot navigation, while determining the exact robot position would be difficult, a human operator could easily joystick the robot to the destination. In the case of human-robot interaction, a user could suggest to the robot how it should have responded to his commands.

Note that we do not assume the oracle and agent maintain the same representation of the world, for example, if both are using POMDPs, they do not necessarily have the same belief state. In fact, it is not necessary for both to be using the same type of representation. At any time, the agent can request an action from the oracle. The oracle will return the optimal action, a_z, for the current time. The agent will then execute the action and use the information to learn as described in the following section.

Representing and Learning Observations

Our primary objective is to learn the observation model assuming the reward and transition functions are known. Thus, $P(s'|s,a)$ and $R(s,a)$ are known, but not $P(z|s)$. The objective is to determine an observation model which replicates the policy of the oracle. It is important to note that this may not be the same observation model necessarily used by the oracle. Different observations functions can result in the same policy, although the policies may have different expected rewards.

We associate with each state a Dirichlet distribution over the possible observations. This acts as a prior over a multinomial which becomes the observation distribution for the state. When a POMDP is created, a multinomial is sampled from the Dirichlet distribution for each state. The resulting multinomial becomes the observation distribution.

Because the environment is partially observable, the exact state is never known so the observation probabilities cannot be updated directly. Instead, the observation distribution of the models is used to adjust the Dirichlet hyper parameters. After quering the oracle, we perform the following update for each model, M_i, which agreed with the oracle:

$$\alpha(s,z) \leftarrow \alpha(s,z) + \lambda f(z) Pr_M(z|s) \quad \forall s \in S, z \in Z \qquad (2)$$

where λ is the learning rate and f is a frequency correction. This update rule acts as a gradient ascent, moving the Dirichlet parameters in the direction of models which have proven to have a policy similar to the oracle's policy. The frequency correction is used to emphasize less frequently occuring actions and is defined as:

$$f(z) = 1 - \frac{\text{\# times z has occured}}{\text{\# total observations}} \qquad (3)$$

This correction helps put more weight on the correct models when a rare action occurs. Otherwise, the system risks being dominated by very frequent actions. This helps avoid local minima by emphasizing all the actions, rather than locating a set of models which only issue the most common action repeatedly. Often times, the selection of the least frequent actions is the most important, such as dialogue system where the system can choose to request more information from the user before determining the final action. In this case, the system may choose to repeatedly request more information from the user as this is the most common action executed by the oracle. Instead, the final action after requesting more information is the important action, but will happen less frequently.

Experiments

The Smartwheeler(Atrash et al. 2009) autonomous wheelchair project aims to develop an autonomous wheelchair for us by patients with mobility issues. The goal of the project is to build a wheelchair which can aid with mobility by removing the physical and cognitive load from users who would have difficulty operating a standard electric wheelchair due to physical impairments, fatigue, or sensory impairments. Our research focuses primarily on the

Figure 1: Autonomous Wheelchair

higher-level interaction and decision-making elements of the system. By shifting autonomy away from the user to the wheelchair, the user can control the system using high-level commands and take advantage of the low level robot control.

An interactive system was designed consisting of several components. A user speaks to a speech recognition system which attempts to transcribe the audio. This transcription is passed to a semantic parser which attaches semantic and grammar information if possible. This annotated information is now handled by the Interaction Manager, which is the decision-making component of the system and will be the focus of these experiments. Feedback is provided to the user through a mounted display. Details of this system are beyond the scope of this paper and can be found in (**?**).

The full wheelchair interaction manager is based on complex POMDP. The parameters for the current system used on the wheelchair are determined by hand labelling much of the training data, as well as data gathered from multiple user tests. Due to computational constraints, for these experiments, we use a simplified model with 7 actions: 6 actions that directly control the robot (move forward, move backward, turn left, turn right, turn around, and stop), as well as one query action requesting additional information. The goal is to learn the observation parameters automatically, as opposed to the model used in the current system which was constructed using labelled data and adjusted by hand.

For these experiments, the handcrafted POMDP acts as the oracle. This model has been constructed using manually adjusted parameters and data accumulated from user tests. The policy for the POMDP is determined using a point-based appoximation method, and the belief state maintained during execution. When the oracle is queried for the policy information, the action for the current belief state of the ground truth POMDP is returned. In a deployed system, these actions would be provided by a user. Using the model as an oracle allows us to run repeated simluation experiments rapidly.

Throughout these experiments, 20 POMDPs are sampled and maintained from the Dirichlet distributions. Every 1000 steps, the three POMDPs with the least likelihood are removed, and three new POMDPs are sampled from the current Dirichlet. Experiments were repeated 10 times, with the average results reported. The initial set of experiments assume the oracle is queried at every step. This allows us to explore the effects of the learning on the system. Reducing the number of queries is addressed later. The prior in this experiment is very roughly approximated by selecting initial counts in proportion to the ground truth distribution. This approximates a situation where a rough model of the world is known. To speed up learning, parameter tying is also applied.

The rate of resampling and the number of models resampled are parameters which can be adjusted for the experiments. Generally, resampling more often and resampling more models results in faster movement through the model space, while sampling less frequently maintains the direction of search.

Because we are interested in the behavior of the resulting POMDPs, the observation parameters values are not compared directly. Instead, a simple evaluation phase is run periodically during the experiment to measure how often the learned models select the same action as the oracle. Beginning with the initial belief state, at every step, an action is selected by the oracle, the action executed, an observation is returned, and the belief state is updated. The action which would have been selected at each step by the sampled POMDPs is compared to the true action. The number on which they agree is tallied. For our experiments, this simulation is run every 1000 iterations, and is run for 1000 steps.

As a baseline for comparison, a model-free method using support vector machines (Mitchell 1997)) was also implemented. Instead of learning a full predictive model of user observations, previous systems have used a model-free supervised learning approach to predict action choices from past user observations. We have applied such a method to our problem. During the learning phase, the history of actions and observations is recorded as a set of (a_t, z_t) pairs. At the beginning of the evaluation, a sliding window is passed over the (a_t, z_t) pairs to create a training set. An SVM is training over these pairs. During evaluation, a recent history $(a_{t-k}, z_{t-k}, ..., a_{t-1}, z_{t-1})$ is maintained. At each step, an action is selected by passing the recent history to the SVM and using the classification as the action. Different history lengths were explored to allow the SVM to look further into the past to determine the action. Shorter windows only provide information about the more immediate context, while longer windows provide more information but make generalization more difficult. Experiments were conducted for different history lengths, $k = 2$ and $k = 4$. Larger window lengths led to computational constraints due to the quadratic programming step involved in SVM training. This method represents a model-free method to learn the policy directly from the actions and observations recorded.

Figure 2 shows the results of learning on this interaction manager using the Bayesian method described (POMDP), as well as the SVM method for different values of k. The results show that learning is occuring using the POMDP method. The SVMs, however, have difficulty learning the domain. This is a result of the nature of the problem. For the task domain used here, looking at a very short history

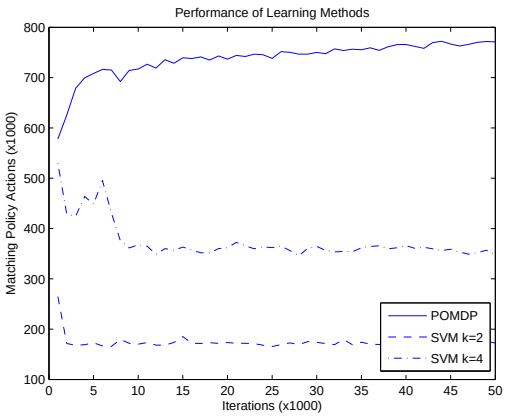

Figure 2: Performance of methods based on Interaction Manager domain.

is sufficient for determining the next action. However, as the model becomes more complex, more history information is needed to determine context. Because the SVM method must be explicitly told the history to consider, it has no way to determine the optimal length. This is further exasperated by the SVMs attempting to generalize over too much information. By considering information too far back in the history, the SVMs are trying to incorporate this irrelevant information into the model, resulting in problems with the classification. The POMDPs, however, maintain history automatically through the use of the belief states and are less affected by this problem.

An example of this occurs because the dialogue system essentially resets itself after each completed action. It is assumed that after one task is complete, the user can request any task. This information is easily encoded into the POMDP through the transition function. A typical dialogue may begin with the user issuing a "turn left" command. The system issues a query, as "turn left" is easily confused with "turn right." The user issues another "turn left" command at which time the system commits to turning left. In another case, the user issues a "stop" command which the system immediately identifies as the correct action, as no other action sound similar. In the first case, the dialogue involved two iterations. In the second case, only one. The POMDPs are able to encode this as one continous input, due to the information in the transition models. However, the SVM approach assumes a fixed history length. Examining a history which is too short may not capture enough of the context to make the correct decision, while a history which is too long captures too much information, looking into a previous iteration which is not relevant to the current action. While this can be handled using domain-specific heuristics, the model-based system is able to capture this information implicity.

There are other advantages to explicitly maintaining a model as opposed to using a supervised learning strategy. Models can be transfered between users, either by starting with the model themselves or converting the model into priors and restarting the learning process. An interaction manager trained on one user can be transfered to another user. While this new model may not necessarily be tuned for the new user, it maybe much closer than a rough hand crafted initial model, especially for elements of the problem which are not user dependant such as noise from the speech recognition. Similarly learning the model allows for adapation if the dynamics of the environment change over time. The preferences of the user may change over time or or, in our target domain, the wheelchair may be moved to a new environment. Modules in the robot might be changed, such as a new microphone or new speech recognition software. By maintaining a model, we can adapt the parameters to accomodate for the changes without disregarding the previously learned information. Finally, the SVM method only provides information about experiences which have been explicitly encountered as direct matches need to be found in the history. By using model, generalization can occur more easily to novel experiences. POMDPs maintain a belief state, which encodes the information from the transition and observation models. These belief states are continuous, so belief states which are never explicitly encountered still have policy information as a result of the way policies are maintained.

RELATED WORK

Learning models in a partially observable environment is a difficult task. Conceptually, the lack of visible state makes it difficult to assign credit when a transition occurs or when an observation is received. The Baum-Welch algorithm (Rabiner 1990) is an expectation-maximization algorithm (Dempster, Laird, and Rubin 1977) traditionally used for learning parameters in hidden Markov models (HMMs) given only a sequence of observations. Hidden Markov models are similar to POMDPs in that they have an unobserved underlying state which must be inferred through a series of observations. The main difference being that HMMs are used for monitoring processes without decision making. Baum-Welch has become a popular method for training HMMs and is used extensively in many applications. However, it tends to require a large amounts of data for training, especially as the models become more complex.

Due to the close nature of HMMs and POMDPs, Baum-Welch can be applied almost directly to learn POMDP parameters, although due to the addition of actions, more data is required for training. Early work (Chrisman 1992) on the application of Baum-Welch to general POMDPs used the same mechanisms of expected counts to re-estimate the parameters iteratively after data is collected. This work considering the idea of a predictive model and also presented mechanisms to adjust the number of underlying states as needed to maximizes the predictive power of the model. Baum-Welch has been used to train POMDPs for robot navigation and localization for the Xavier robot (Koenig and Simmons 1996). Here, POMDPs were used to represent the physical layout of a building. The transition and observation parameters were learned based on traces of the robot moving through the environment using sliding window of history over which the parameters were refined at intervals. The topology of the environment was learned by maintain-

ing several potential distances of each hallway over which the model is learned. Baum Welch was again used to learn navigation POMDPs (Shatkey and Kaelbling 1997), this time with less constraints on than topology. Both of these works took advantage of the domain to aid in learning.

Active learning (Cohn, Ghahramani, and Jordan 1995) has also been explored as a method for learning probabilistic models. Under active learning, the system itself guides the learning by determining which samples provide the most information using heuristics such as predicted variance. Efficient algorithms for active learning in HMMs have been presented (Anderson and Moore 2005). In this work, samples are selected to minimize model uncertainty, as well as considering the cost of misclassification. Recently work has extended active learning to POMDPs. The MEDUSA algorithm (Jaulmes, Pineau, and Precup 2005) uses an oracle which provides state information to update a Bayesian prior over model parameters. Queries are issued based on a set of heuristics which examine the current state of the models as well as potential information gain from the query. Similar work (Doshi, Pineau, and Roy 2008) also relies on an oracle to provide information, but instead learns a the reward function of the user. Queries are issued based on minimizing the Bayes-risk of an action. Many of these heuristics are easily incorporated into our framework.

DISCUSSION AND FUTURE WORK

We have demonstrated an algorithm for learning the observation function of a probablistic model based on the feedback from an oracle. We have shown that by maintaining a distribution over the potential parameters and updating that distribution based on feedback from the user, more accurate models can be sampled over time within a Bayesian reinforcement learning framework. Finally, we have demonstrated this algorithm on a basic robot interface domain.

This paper focused specifically on learning the observation parameters. However, it would be straightforward to extend the same learning mechanism to learn the transition parameters as well. The current work assumes the system dynamics are known. In the case of an interaction manager, much of this comes from the frequency of the commands used by the user, as well as the order in which different commands are typically requested. Ideally, the agent would be able to learn these parameters as well as the observations and rewards. We would like to continue this direction and integrate our previous work which learned the reward parameters as well, resulting in a system which is able to learn the complete model.

References

Anderson, B., and Moore, A. 2005. Active learning for hidden Markov models: Objective functions and algorithms. In *International Conference on Machine Learning*, 9–16.

Atkeson, C., and Schaal, S. 1997. Robot learning from demonstration. In *International Conference on Machine Learning*, 12–20.

Atrash, A., and Pineau, J. 2009. A Bayesian reinforcement learning approach for customizing human-robot interfaces. In *International Conference on Intelligent User Interfaces*, 355–360.

Atrash, A.; Kaplow, R.; Villemure, J.; West, R.; Yamani, H.; and Pineau., J. 2009. Development and validation of a robust speech interface for improved human-robot interaction. *International Journal of Social Robotics* 1:345–356.

Chrisman, L. 1992. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *National Conference on Artificial Intelligence*, 183–188.

Cohn, D.; Ghahramani, Z.; and Jordan, M. 1995. Active learning with statistical models. In *Advances in Neural Information Processing Systems*, 705–712.

Dearden, R.; Friedman, N.; and Andre, D. 1999. Model based Bayesian exploration. In *Uncertainty in Artifical Intelligence*, 150–159.

Dempster, A.; Laird, N.; and Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society* 39:1–38.

Doshi, F.; Pineau, J.; and Roy, N. 2008. Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs. In *International Symposium on Artificial Intelligence and Mathematics*.

Duff, M. 2002. *Optimal learning: Computational procedures for bayes-adaptive markov decision processes*. Ph.D. Dissertation. Director-Andrew Barto.

Jaulmes, R.; Pineau, J.; and Precup, D. 2005. Active learning in partially observable Markov decision processes. In *European Conference on Machine Learning*.

Kaelbling, L.; Littman, M.; and Cassandra, A. 1998. Planning and acting in partially observable stochastic domains. In *Artificial Intelligence*, 99–134.

Koenig, S., and Simmons, R. 1996. Unsupervised learning of probabilistic models for robot navigation. In *International Conference on Robotics and Automation*.

Mitchell, T. M. 1997. *Machine Learning*. New York: McGraw-Hill.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, 1025–1032.

Poupart, P.; Vlassis, N.; Hoey, J.; and Regan, K. 2006. An analytic solution to discrete Bayesian reinforcement learning. *National Conference on Artificial Intelligence* 1.

Rabiner, L. R. 1990. A tutorial on hidden Markov models and selected applications in speech recognition. In *Readings in Speech Recognition*, 267–296.

Ross, S.; Chaib-draa, B.; and Pineau, J. 2007. Bayes-adaptive POMDPs. In *Neural Information Processing Systems*.

Shatkey, H., and Kaelbling, L. 1997. Learning topological maps with weak local odometric information. In *International Joint Conference on Artificial Intelligence*, 920–929.

Spaan, M., and Vlassis, N. 2005. Perseus: Randomized point-based value iteration for POMDPs. In *Journal of Artificial Intelligence Research*, 195–220.