Learning Probabilistic Models for Decision-Theoretic Navigation of Mobile Robots

Daniel Nikovski DANIELN@CS.CMU.EDU
Illah Nourbakhsh ILLAH@CS.CMU.EDU
The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213 USA

Abstract

Decision-theoretic reasoning and planning algorithms are increasingly being used for mobile robot navigation, due to the significant uncertainty accompanying the robots' perception and action. Such algorithms require detailed probabilistic models of the environment of the robot and it is very desirable to automate the process of compiling such models by means of autonomous learning algorithms. This paper compares experimentally four learning methods in combination with four heuristic decision-theoretic planning algorithms for the purpose of learning a probabilistic model of the environment of a mobile robot and using this model for navigation. One of the learning methods is novel and presents an approach to probabilistic model learning based on merging states by clustering trajectories of observation/action pairs. The strengths and weaknesses of each combination of learning and planning method is explored in a sample environment for mobile robot navigation.

1. Introduction

Mobile robots operating in office environments have to make decisions under significant uncertainty in their observations and the effect of their actions. Furthermore, sometimes a set of locations in these environments are indistinguishable from one another, which introduces perceptual aliasing and hidden state. Decision-theoretic planning (Boutilier et al., 1999) has become the preferred approach to robot control in such situations and has been implemented on several research prototypes of mobile robots (Nourbakhsh, 1998; Koenig & Simmons, 1998). A decision-theoretic planner is usually provided with a stochastic model of the environment, which reflects the uncertainty in the robot's sensing and actions. Hand-crafting such a model is typically time-consuming and error-prone and might require blueprints and/or CAD models of the buildings the robot must operate in, as well as considerable human effort. Hence, it is very desirable to develop methods for learning such stochastic models from observation/action traces obtained from autonomous exploration of the environment on the part of the robot.

This paper reports experimental results from simulated exploration and control of an office mobile robot, which uses four learning methods to acquire stochastic models of the environment it operates in and applies four planners to the learned models in order to navigate the robot to its home position. One of these learning methods is novel and explores a state-merging approach to model learning based on clustering percept/action trajectories, in contrast with the traditional methods based on nonlinear maximization of observation likelihood. The principal goal of the reported experiments was to determine experimentally which combination of learning and planning methods works best for mobile robot navigation in office spaces and explore the limitations of these approaches. Section 2 provides an overview of decision-theoretic methods for representing domains, reasoning, and planning by means of partially-observable Markov decision processes (POMDPs). Section 3 presents four algorithms for learning POMDPs from data. Section 4 describes the experimental environment, and experimental results are reported in section 5. Section 6 discusses the results and some directions for future work.

2. Decision-theoretic planning

Decision-theoretic planning is an extension of the classical AI planning paradigm and can handle problems in which the effect of actions is uncertain and observations are noisy and incomplete. It is based on the frameworks of Markov decision processes (MDPs) and partially-observable MDPs (POMDPs).

2.1 Representation and reasoning in POMDPs

A POMDP is described by the tuple (S, π, A, T, O, E, R), where S is a set of states, π is an initial probability distribution over these states, A is a set of actions, and T is a transition function that maps $S \times A$ into discrete probability distributions over S. The states in S are not observable – instead, observations from the set O can be perceived only. The function E maps $S \times A$ into discrete probability distributions over O (in some POMDPs the observations depend on state only). The function R describes the immediate reward received when the POMDP is in each of the states in S.

The structure, transition, emission, and reward functions are based on the control objectives of the agent that is controlling the POMDP. The scenario we are interested in is mobile robot navigation, where each state is a location in a world and each observation is a discrete percept that can be produced by the perceptual apparatus of a mobile robot. If the objective of the robot is to reach a particular goal state and stay there, the reward for that state can be, for example, one, while the reward for being in all other states can be zero. Then, if the agent maximizes the reward it receives, it will effectively be achieving its goal.

A POMDP can be controlled by executing one of the available actions at each time step. As a result, the POMDP transfers to a new unobservable state and emits a new observation which can be perceived by the agent that is controlling the POMDP. The agent has to infer the state of the POMDP on the basis of the sequence of observations and the knowledge it has about the transition and emission probabilities of the POMDP. Since the agent has to reason under uncertainty, it cannot be completely sure about the exact state the POMDP is in; instead, it has to maintain a *belief state* $Bel(S)$ represented as a probability distribution over all states in S (Cassandra et al., 1996; Boutilier et al., 1999).

2.2 Planning with POMDPs

Controlling a POMDP amounts to choosing the correct actions that will maximize the expected cumulative reward received by the agent over the course of its operation. When this course is infinite in duration, the sum itself will be infinite – one way to avoid this is to apply an exponential discounting factor $\gamma < 1$ to each reward. Thus, the goal of the controller is to maximize the quantity $< \sum_{t=0}^{\infty} \gamma^t R_t >$, where R_t is the reward received at the t-th step of operation, and $< \cdot >$ denotes expectation.

Several algorithms exist for finding the optimal control policy in POMDPs (Kaelbling et al., 1996). However, since this problem has been proven to be PSPACE hard, we are considering approximate solutions such as assumptive planning and MDP-based approaches (Cassandra et al., 1996; Nourbakhsh, 1998; Koenig & Simmons, 1998).

Assumptive planning is a heuristic strategy proposed by Nourbakhsh (1998), which performs full belief updating of its belief state based on the transition and emission probabilities, but makes several simplifying assumptions when choosing an action. First, it assumes that it is in the most likely state with complete certainty, thus ignoring the possibility of being in other states. Next, it constructs a deterministic FSA from the transition and emission probabilities of the original POMDP and uses a general search algorithm such as iterative deepening to find a path in the FSA. Furthermore, the planner produces a list of percepts that ought to be seen if the plan is executed and the FSA is a true representation of the world. However, in most cases the percepts seen by the agent will differ from the expected ones, which is an indication that the plan is no longer valid and the agent is lost. If this is the case, the planner is called again to find a new plan based on the latest estimate of the most likely state.

Another set of heuristic strategies solves first the underlying MDP and then makes use of that solution in conjunction with the estimated belief state (Cassandra et al., 1996; Koenig & Simmons, 1998). The solution of the underlying MDP is an optimal policy that maps each state into the action that maximizes the expected future cumulative reward. In order to find that action, an auxiliary function $Q(s, a)$ is computed, whose meaning is the expected cumulative reward if action a is performed in state s and an optimal policy is followed thereafter. If the Q-function is known for a state, the optimal action a^* for that state is the one with highest Q-value: $a^*(s) = argmax_a Q(s, a)$. The Q-function can be found by means of Q-learning, an iterative Monte-Carlo reinforcement learning method (Sutton & Barto, 1998):

$$\hat{Q}^{t+1}(s, a) := (1-\eta)\hat{Q}^t(s, a) + \eta[R(s) + \gamma \max_b \hat{Q}^t(s', b)],$$

where $\hat{Q}^t(s, a)$ is an estimate of the Q-function at iteration t, η is a learning rate coefficient, which should decrease gradually with time, and the state s' is sampled according to the transition probabilities for state s and action a. Once the MDP is solved, the Q-values can be used in several ways to approximate the optimal policy for the POMDP case (Cassandra et al., 1996; Koenig & Simmons, 1998). The *most-likely-state method* (MLS) chooses the optimal action for the most likely state, ignoring the possibility that the robot might be in other states. The *voting method* chooses the action with highest probability mass in the belief vector. The Q_{MDP} *method* is similar to the voting method, but instead of adding up state beliefs only to the sum of the winning action, they are added to the sums of all actions, weighted proportionally to the actual Q-values.

3. Learning POMDPs

Learning a POMDP from observations instead of having one available in advance furthermore complicates the problem of determining optimal policies. Several fundamental questions arise, among which the correspondence between world and model states, and the related problem of goal determination. In general, the agent does not know how many states existed in the original process that generated the data and has to make a decision about the final number of states in the model. Another difficulty is that of determining the goal criteria for planning with the learned POMDP. Both assumptive planning and the MDP-based strategies need a goal state – either to terminate the iterative deepening search for assumptive planning, or to construct a proper reward function for the MDP-based algorithms. However, when a POMDP is learned, it is not clear which of its states correspond to the true goal states in the real world; in general, there won't be one-to-one correspondence between learned and true states at all. This circumstance changes significantly the goal criterion used for planning.

One possibility is to transfer the goal from the state domain to the perceptual domain, that is, instead of trying to reach a goal location, the agent tries to observe the percept that corresponded to that location in the training sensory-motor trace. For assumptive planning, the states of the FSA that is an idealized representation of the learned POMDP can be labeled with the most likely percept to be observed in that state. A solution exists for the MDP-based planners as well. Instead of assigning reward one to the goal state and zero to all other states, the reward for each state can be equal to the probability that the goal percept will be observed in this state. Thus, a policy that maximizes reward will in effect maximize the probability that the goal percept is seen; if the system is at the goal location each time the goal percept is seen, this policy will also maximize the probability of reaching the goal state.

Learning in POMDPs can be performed by means of general algorithms for learning probabilistic networks from data, such as those proposed in (Russell et al., 1994). A POMDP can be represented as a chain of time slices, each of which has one state node and one observation node; all time slices use the same transition and emission matrices. The objective of learning is to find values for the entries in the transition and emission conditional probability tables (CPT) of the POMDP, which maximize the log-likelihood $\ln P(\mathbf{D}|\mathbf{w})$ that the training data set $\mathbf{D}$ was generated by the POMDP with parameters $\mathbf{w}$. Each case D_l from the data set $\mathbf{D}$ consists of assignments for the observable nodes in O.

3.1 Learning POMDPs by iterative adjustments of probabilities

The traditional approach to learning POMDPs has been to fix the structure of the POMDP, initialize the probability matrices of that structure, and iteratively adjust them so as to maximize the likelihood that the training data was generated by the model. Two such algorithms have been considered in our experiments: the first one is the Steepest Gradient Ascent algorithm due to Russell et al. (1994), and the other one is the Baum-Welch learning rule, based on the expectation-maximization (EM) algorithm.

3.1.1 Gradient ascent in likelihood

Russell et al. (1994) proposed a particularly simple learning rule which performs gradient ascent in the space of CPT entries:

$$\Delta w_{ijk} = \eta \sum_l \frac{\partial P(D_l|\mathbf{w})/\partial w_{ijk}}{P(D_l|\mathbf{w})} = \sum_l \frac{P(S_{ij}, \mathbf{U}_{ik}|D_l, \mathbf{w})}{w_{ijk}},$$

where w_{ijk} is the entry of the CPT of state node S_i that designates the probability that this node S_i will be in its j-th state s_{ij} given that its parents $\mathbf{U}_i$ are in their k-th configuration $\mathbf{u}_k$ (Russell et al., 1994). The summations index l runs over all data cases D_l in $\mathbf{D}$.

3.1.2 Baum-Welch

Chrisman (1992) and Koenig and Simmons (1998) adapted the Baum-Welch algorithm for learning hidden Markov models (HMMs) to the problem of improving the entries of the CPTs of POMDPs from data. The learning rule is

$$\hat{w}_{ijk}^{T+1} = \frac{\sum_l \xi_{i,l}^T(j,k)}{\sum_l \gamma_{i,l}^T(j)},$$

where $\hat{w}_{ijk}^{T+1}$ is the estimate of the transition probability $P[S_i(t+1) = s_{ik}|\mathbf{U}_i(t) = \mathbf{u}_{ij}]$ that node S_i will be in its j-th state s_{ij} given that its parents $\mathbf{U}_i$ are in their k-th configuration $\mathbf{u}_k$, after iteration $T+1$, and

$$\xi_{i,l}^T(j,k) = P[S_i(t+1) = s_{ik}, \mathbf{U}_i(t) = \mathbf{u}_{ij} | D_l, \hat{\mathbf{w}}^k]$$

$$\gamma_{i,l}^T(j) = P[\mathbf{U}_i(t) = \mathbf{u}_{ij} | D_l, \hat{\mathbf{w}}^k].$$

3.2 Learning POMDPs by state merging

Baum-Welch and Steepest Gradient Ascent can be used to learn HMMs and POMDPs, but in general do not converge to the true models that generated the training data (Nikovski, 1998). Both algorithms are iterative – they start with a random assignment of parameters and iteratively update them in order to maximize the likelihood of the data. Very often they fail to converge to transition matrices that are close to deterministic. One possible reason is that the probabilities in these transition matrices have random initial values and it is highly unlikely that the learning algorithms will overcome local maxima in likelihood to drive the parameters to the boundaries of parameter space, which correspond to close to deterministic transition functions. Furthermore, failing to converge to such functions can have negative impact on heuristic methods such as assumptive planning, which simplify the POMDP into an FSA and can be exptected to perform best in comparison with optimal POMDP planning algorithms if the loss of precision due to the simplification is minimal.

In an attempt to overcome these shortcomings of Baum-Welch and Steepest Gradient Ascent, two novel algorithms for learning POMDPs were explored, both of which based on state merging.

3.2.1 Best-first model merging

A completely different method for learning HMMs has been suggested by Stolcke and Omohundro, originally for the purposes of speech recognition (Stolcke & Omohundro, 1993). The algorithm builds an initial model, which has a state for each observation/action pair. We will call the states in this initial model *time-states* to distinguish them from the true states of the POMDP, which generated the observation data. This model fits the training data perfectly and has the highest possible log-likelihood, but cannot generalize over novel cases, because in practice it overfits the data.

The objective of the algorithm is to find which time-states came from the same true POMDP state and reduce the initial overfit model to the correct one. This is done by consecutive merging of pairs of time-states that are likely to correspond to the same hidden state in the underlying POMDP that generated the data. The criterion for merging is the resulting decrease in data likelihood. At each step of the algorithm, several candidate mergers are computed, the one that decreases the likelihood the least is accepted, and the search continues in a greedy fashion until the likelihood of the model with respect to the data becomes unacceptably low, or a predetermined number of states is reached. This algorithm was adapted by us for the purpose of learning POMDPs, much like Chrisman (1992) adapted Baum-Welch for the same purpose. Experimental comparisons with Baum-Welch on 14 synthetic worlds, reported in (Nikovski & Nourbakhsh, 1999), demonstrate that there are many cases when the best-first model merging (BFMM) algorithm outperforms Baum-Welch significantly. However, the computational complexity of BFMM is $O(N^3)$, where N is the number of observations, and is by far the slowest of all algorithms considered in our experiments.

3.2.2 State merging by trajectory clustering

Another major shortcoming of the BFMM algorithm is that state merging proceeds greedily and never reconsiders suboptimal mergers of pairs of time-states. A better approach would be to rank somehow all possible mergers, and actually carry out only the most promising ones.

The ultimate objective of state merging is to group the time-states for all data points into several groups such that the time-states in a single group are likely to correspond to the same state of the true POMDP that generated the data. This observation suggests the idea to do clustering of the time-states based on some form of similarity between them. The simplest approach is to merge the time-states, which have emitted the same symbol, but it is bound to fail when perceptual aliasing is present. A much better measure of similarity is the length of matching action/observation sequences prior to the two time-states. This is exactly the similarity measure that McCallum used in his instance-based Q-learning algorithm (McCallum, 1995). The intuition behind this measure is that the trajectories leading to a time-state represent an embedding space of the true hidden state space, and close points in the embedding space (matching trajectories) correspond to close hidden states. This is a common approach used in system identification, which exploits the fact that while the immediate observation is not a reliable indication of the true state of the system, a whole sequence (trajectory) of such observations usually disambiguates the state. Furthermore, McCallum's algorithm was constrained by design only to matching trajectories leading *into* the current time-state, while our novel algorithm has available data after the current moment and can also match sequences leading *out of* the current time-state. So, an analogous measure is the length of matching sequence after the two time-states. The lengths of both sequences (leading into and out of the current time-state) can be added up as well. We will denote henceforth the length of the matching trajectories prior to the two states with b, and the length of those after the two states with f. We will also introduce the variable c, which has a value of 1 if the two directly emitted symbols from the two states match, and 0 if they don't. In our experiments we used the sums of all possible combinations of the three measures b, f, and c.

The algorithm executes in the following four steps:

1. Compute similarities between all possible pairs of time-states and place them in a similarity matrix.

2. Perform clustering of the time-states based on this similarity matrix. The time-states in the same cluster are assumed to correspond to the same true state of the POMDP.

3. Label the time-states with the number of their respective cluster from the previous step. At this point, the POMDP becomes a fully-observable MDP (FOMDP). (However, some of the states might possibly be mislabeled).

4. Compute the transition and emission matrices of the FOMDP – this is now straightforward.

It should be noted that some of the more popular clustering algorithms such as k-means cannot be applied to the second step of the algorithm, because they require averaging of data points, while in this case there is no underlying metric space in which addition and multiplication are defined. Nevertheless, there are clustering algorithms that can work with a similarity matrix alone. One such algorithm, widely used in pattern recognition, is based on finding minimum spanning trees (MST) in the graph whose adjacency structure is defined by the similarity matrix (Duda & Hart, 1973). Once the MST is found, the edges corresponding to the *least* similar pairings are severed. This results in several cliques, which define clusters of time-states that are likely to correspond to the same state in the true POMDP that generated the observations. The last step of this algorithm for state merging by trajectory clustering (SMTC) is to assign consecutive numbers to the remaining cliques, label the hidden states in the observation sequence with their respective clique numbers, and estimate the transition and emission probabilities of the POMDP as if it was fully observable.

This approach is likely to give better results than BFMM, because it considers all possible mergers before actually carrying out any of them. Furthermore, this method is guaranteed to recover completely a fully observable POMDP, while this is not necessarily true for SGA, EM, and BFMM. (SGA and EM might get stuck in a local maximum of log-likelihood, while BFMM might choose a very bad merger due to the limited number of mergers it considers at each step.) Thus, it can be expected that SMTC will outperform EM and BFMM in worlds, which are mildly unobservable, such as environments with moderate perceptual aliasing.

Just like with BFMM, this algorithm is quite expensive computationally, if implemented directly. Finding the similarity matrix in a sequence of N actions/observations has a worst-case running time of $O(N^3)$, because N^2 matches are considered, and the length of the matching subsequences backwards and forwards can be as long as N elements. In practice, though, most matches will terminate after few time steps. Still, the complexity of finding the MST given the similarity matrix of N^2 elements is $O(N^3)$, if Prim's algorithm is used, and $O(N^2 \log N)$, if Kruskal's algorithm is employed in combination with a fast sorting routine. Finally, the clique-labeling stage takes computations in the order of N^3, if matrix multiplications are used to determine adjacency between members within a clique.

4. Experimental environment

The experiments reported below used a commercial simulator of Nomad 150 robots, available from Nomadic Technologies, Mountain View, California. The Nomad 150 robot is equipped with 16 infrared sensors, which give proximity readings in the range of $0 - 36$ inches. The readings have relatively low noise and high repeatability, but because of their limited range, the robot often experiences significant perceptual aliasing.

The experimental world shown in Fig.1 illustrates this problem. The size of the open space (white) surrounded by the obstacles (black) is 100 by 50 inches, and the robot starts exploration at coordinates $(25, 25)$ inches, with the origin of the coordinate system at the lower left corner of the open space. Three actions are allowed: move forward 25 inches, turn left 90 degrees, and turn right 90 degrees. If the robot cannot complete a whole move of 25 inches because of a collision with a wall, it backs up to its original position.

Thus, the robot can be at one of three locations, and can have one of four orientations (plus some small random drift supplied by the simulator, which does not result in more than 0.5 inches difference from these locations over a course of 200 steps). Thus, the robot can be in one of 12 different states at any time, and perceive a 16-dimensional vector of infrared readings. The fact that this world has only 12 discrete states makes it similar to the maze worlds commonly used in reinforcement learning research, with the difference that the observations here are continuous, and not discrete. The size of the state space is comparable to that used in previous work on recovering worlds with hidden state – Chrisman (1992) experimented with a space-station docking problem with 6 states, and McCallum (1995) used synthetic worlds with 8, 11, 14, and 15 states. All of these previous experiments used discrete observations.

Some of the states in this world are indistinguishable from each other – the following two pairs of states generate the same readings: $(50, 25, 0)$ and $(50, 25, 180)$; $(50, 25, 90)$ and $(50, 25, 270)$. This is due to the limited range of the infrared sensors (36 inches at most), which does not allow the robot to perceive the distinguishing feature in the upper right corner of the area when it is at the central location with coordinates $(50, 25)$. However, the leftmost and rightmost locations are identifiable from each other because of this feature. This is one type of aliasing typically arising in office spaces when the robot is in a long corridors.

A total of 200 observation/action pairs were acquired during the exploration stage in one sequence, with the robot starting at the leftmost location facing east $(25, 25, 0)$. The 16-dimensional vectors of continuous observations were quantized into 12 symbols by k-means clustering – between 5 and 20 iterations were typically necessary for convergence. After the observation vectors were quantized, each of the observation/action pairs were labeled with one of these 12 symbols, which were used in the emission tables of the learned POMDP models. Note that at least some of the symbols labeled no states at all, because the number of discernible states is less than their total number, due to perceptual aliasing. The correct number of states is given to the learning algorithms and they have to learn the transition and emission probabilities of the POMDP. Even though each action will be tried on average 6 times in each state, it is often the case that some state/action pairs are never experienced. As a result, some of the learning algorithms sometimes produce transition matrices, which are not strictly stochastic, i.e. the sum of all probabilities out of a certain state for a particular action might be 0 instead of the required value of 1 for stochastic matrices. However, the planning algorithms deal with such cases in a straightforward manner – assumptive planning never considers such actions in the planning process, as if this action was not available in this state, and the iteration of the Q-learning algorithm of the MDP-based planners assigns zero Q-value to this state/action pair. This is a reasonable and practical approach and it can be expected that no other learning and planning algorithms can do better in cases when certain transitions are simply not present in the training data.

The goal of the robot was to reach the home location $(25, 25)$, facing east (steering angle 0). After a robot acquired a model by means of one of the four learning methods described above, it was placed at one of the 12 available starting locations and controlled by one of the four planning methods. If it reached the goal within 10 action steps, it was given a discounted reward equal to 0.9 raised to the number of steps it took to reach the goal, following the methodology of (Cassandra et al., 1996). Conversely, if it failed to reach the goal, it was given a reward of zero. For a particular planner and learning method, the rewards were averaged over the twelve starting states, and since one of the states was a goal state and no actions were necessary, any combination of learning method and planner is guaranteed a reward of at least 1/12. The average cumulative discounted award for random action selection was computed as well, to be used as a comparison baseline – any combination of learner/planner can be claimed to have built a useful POMDP model only if it achieves significantly better cumulative discounted reward than that corresponding to random choice of actions.

The MDP-based planners solved the underlying MDP of the learned POMDP by Q-learning, sweeping each state in turn and sampling successive states according to the transition probabilities of the MDP. Learning rate $\eta = 0.1$ and discounting factor $\gamma = 0.9$ were used, for a total of 1000 sweeps.

5. Experimental results

The experimental results are shown in Table 1 for four learning algorithms, the last of which, SMTC, had seven modifications based on the similarity measure used in clustering trajectories. The similarity measure was a sum of one or more of three components: c, which was 1 or 0 depending on whether the states being matched emitted exactly the same symbol; b, the maximum length of matching action/observation pairs *prior* to the two states; and f, the maximum length of matching action/observation pairs *after* to the two states. It should be noted that if $b > 0$ then necessarily $c = 1$. The components present in the similarity measure for a particular modification of the SMTC algorithm are shown in the name of that modification in Table 1; for example, SMTCbf means that the similarity measure used was $b + f$.

Each entry in Table 1 corresponds to one combination of learning and planning method and is of the form $\mu \pm s/z$, where μ is the average cumulative discounted reward achieved over 5 runs, s is the sample standard deviation of that reward, and z is the z statistic measuring the number of standard deviations between μ and the average award μ_r achieved by random number selection over 55 runs (5 for each of the 11 learning methods listed in the table).

The z tests, computed in Table 1, make the assumption of Gaussian distributions of the respective samples. Hence, the computed z values have to be regarded with some caution, because while the distribution of rewards for random choice of actions is indeed roughly Gaussian, the distribution of rewards for a pair of a learner and a planner is most typically not. The usual behavior of the robot during a series of test runs is either to go to the goal directly and achieve maximum reward, if a good model has been acquired, or to bang obstinately into a wall and get no reward at all. This is due to the fact that the MDP-based policies and assumptive plans are fixed, and since the belief distribution converges to a fixed value when the robot is held at the same location, the POMDP policies are in practice fixed as well and if they are wrong, the robot cannot escape that location. Consequently, the rewards are distributed towards the extremes, which violates the Gaussian requirement for z-tests.

With this cautionary note in mind, the results from Table 1 can be interpreted to indicate that the following combinations of learning and planning methods achieve performance significantly better than random action selection: SGA/AP, BW/AP, SMTCc with all MDP-based planners, and SMTCcbf with all planners. The version of SMTC, which employs the sum of all matching components (SMTCcbf) performs best, which confirms expectations. However, those versions of SMTC, which use the backward matching distance b, but not the direct match c, perform very badly, even though if $b > 0$, $c = 1$, as mentioned above.

For the sake of comparison, the last line of Table 1 lists the performance of the four planners when they have a fully observable model of the world (still represented as a POMDP, but each of whose states always emits the same unique observation). It is evident that all planners always find the optimal plan in this world, which suggests that whenever a combination of a learning and a planning method achieves suboptimal results, this is most likely due to acquiring a wrong model on the part of the learner rather than on failure to use the model correctly. It can be seen that while at least some of the learning methods achieve performance statistically significantly better than random action selection, they are still far from recovering reliably a POMDP model of even such a simple test environment.

6. Conclusions and future work

Several algorithms for autonomous learning of POMDP models from data were tested in conjunction with four approximate decision-theoretic planners on a simplified robot navigation task. One of these methods, SMTC, is novel and is based on a radically different approach than existing methods, which typically employ nonlinear optimization of log-likelihood. The SMTC algorithm clusters states based on the similarities between trajectories that lead into and out of the states and subsequently compiles POMDPs based on the formed clusters.

Several similarity measures for state clustering were explored and the sum of matching sequences into and out of the matched states proved to be best not only among the other similarity measures, but also achieved the best performance among all tested algorithms. Still, the algorithm fails to recover the correct POMDP quite often and new improved similarity measures should be explored.

Another major challenge is to scale the algorithm to truly continuous state spaces, since the current simplified world is in practice discrete, as most experimental worlds in reinforcement learning research. This leads to the associated problem of finding a small number of actions, sequences of which are guaranteed to reach any goal state. It is essential to keep the number of actions small, because their number affects adversely the computational complexity of the algorithms.

The current algorithmic implementations of the algorithms should be improved as well, because the number of observations used in the experiments, $N = 200$, is a practical limit for the time being – SMTC takes 63 seconds implemented in Matlab and C++ (MEX files) on a Pentium II computer running at 350MHz. Both BFMM and SMTC have computational complexity $O(N^3)$ and do not allow experiments with much longer observation sequences, which would be absolutely necessary for larger worlds. However, we are currently working on a new version of SMTC, which employs a fast matching algorithm from the field of DNA analysis and expect to achieve complexity of $O(N \log N)$, which would allow experiments with much larger worlds.

Table 1. Results for four learning methods and five planners. Shown are the average reward over five trials, the associated standard deviation, and the statistical z test for difference between the achieved reward and that of random action selection. Abbreviations: AP – assumptive planning; MLS – most-likely state MDP-based; Voting – voting MDP-based; Q_{MDP} – MDP-based proportional to Q-values. See the text for the definitions of the variables c, b, and f.

Method	AP	MLS	Voting	Q_{MDP}	Random
SGA	$0.32 \pm 0.06/+4.13$	$0.25 \pm 0.11/-1.65$	$0.26 \pm 0.11/-0.63$	$0.26 \pm 0.11/-0.63$	0.269 ± 0.056
BW	$0.32 \pm 0.06/+4.13$	$0.25 \pm 0.11/-1.65$	$0.26 \pm 0.11/-0.63$	$0.26 \pm 0.11/-0.63$	0.269 ± 0.056
BFMM	$0.26 \pm 0.16/-0.40$	$0.23 \pm 0.14/-2.52$	$0.23 \pm 0.14/-2.52$	$0.26 \pm 0.16/-0.40$	0.269 ± 0.056
SMTCc	$0.27 \pm 0.14/+0.32$	$0.33 \pm 0.11/+4.23$	$0.33 \pm 0.11/+4.23$	$0.33 \pm 0.11/+4.23$	0.269 ± 0.056
SMTCb	$0.22 \pm 0.08/-4.07$	$0.17 \pm 0.12/-7.31$	$0.17 \pm 0.12/-7.31$	$0.17 \pm 0.12/-7.31$	0.269 ± 0.056
SMTCf	$0.27 \pm 0.08/+0.28$	$0.23 \pm 0.18/-2.29$	$0.25 \pm 0.17/-1.46$	$0.23 \pm 0.18/-2.37$	0.269 ± 0.056
SMTCcb	$0.28 \pm 0.00/+0.85$	$0.28 \pm 0.00/+0.85$	$0.28 \pm 0.00/+0.85$	$0.28 \pm 0.00/+0.85$	0.269 ± 0.056
SMTCcf	$0.28 \pm 0.01/+0.63$	$0.26 \pm 0.06/-0.61$	$0.26 \pm 0.06/-0.61$	$0.26 \pm 0.06/-0.61$	0.269 ± 0.056
SMTCbf	$0.20 \pm 0.11/-5.17$	$0.21 \pm 0.09/-4.27$	$0.21 \pm 0.09/-4.27$	$0.21 \pm 0.09/-4.27$	0.269 ± 0.056
SMTCcbf	$0.33 \pm 0.07/+4.94$	$0.33 \pm 0.07/+4.94$	$0.33 \pm 0.07/+4.94$	$0.33 \pm 0.07/+4.94$	0.269 ± 0.056
True	$0.65 \pm 0.00/+32.88$	$0.65 \pm 0.00/+32.88$	$0.65 \pm 0.00/+32.88$	$0.65 \pm 0.00/+32.88$	0.269 ± 0.056

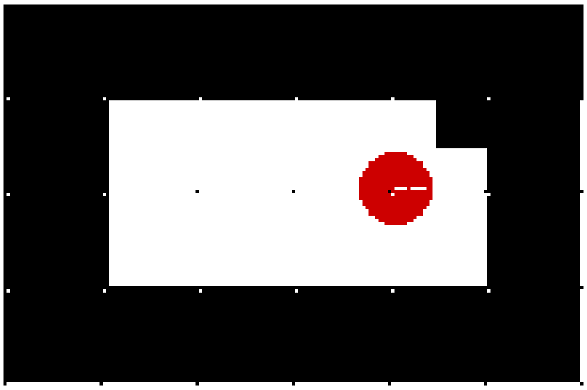

Figure 1. Experimental world for learning and planning. The open space has a size of 100 by 50 inches, which results in a total of 12 possible location/orientation pairs.

References

Boutilier, C., Dean, T., & Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, *11*, 1–94.

Cassandra, A. R., Kaelbling, L. P., & Kurien, J. A. (1996). Acting under uncertainty: Discrete Bayesian models for mobile robot navigation. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems.*

Chrisman, L. (1992). Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 183–188). San Jose, CA: MIT Press.

Duda, R., & Hart, P. (1973). *Pattern recognition and scene analysis.* John Wiley and Sons.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1996). *Planning and acting in partially observable stochastic domains* (Technical Report CS-96-08). Brown University, Providence, RI.

Koenig, S., & Simmons, R. (1998). Xavier: a robot navigation architecture based on partially observable Markov decision process models. In D. Kortenkamp, R. Bonasso and R. Murphy (Eds.), *Artificial intelligence and mobile robots*, 91–122. Cambridge: MIT Press.

McCallum, R. A. (1995). Instance-based state identification for reinforcement learning. *Advances in Neural Information Processing Systems* (pp. 377–384). The MIT Press.

Nikovski, D. (1998). Learning stationary temporal probabilistic networks. In *Proceedings of the Conference on Autonomous Learning and Discovery.* Pittsburgh, PA: Carnegie Mellon University.

Nikovski, D., & Nourbakhsh, I. (1999). Learning discrete Bayesian models for autonomous agent navigation. In *Proceedings of the IEEE Symposium on Computational Intelligence in Robotics and Automation.*, (pp.137–143). Monterey, CA: IEEE.

Nourbakhsh, I. (1998). Dervish: an office navigating robot. In D. Kortenkamp, R. Bonasso and R. Murphy (Eds.), *Artificial intelligence and mobile robots*, 73–90. Cambridge: MIT Press.

Russell, S., Binder, J., & Koller, D. (1994). *Adaptive probabilistic networks*Technical Report CSD-94-824). University of California, Berkeley.

Stolcke, A., & Omohundro, S. (1993). Hidden Markov Model induction by Bayesian model merging. *Advances in Neural Information Processing Systems* (pp. 11–18). Morgan Kaufmann, San Mateo, CA.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning.* Cambridge: The MIT Press.