Varying Ability to Observe in a Partially Observable World

CS394R - Project Report

Aishwarya Padmakumar
UT-EID: ap44694

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) are a framework suitable for the formulation of decision making problems in the presence of noisy sensors and actuators such as those of real world agents [9]. However, in many applications, the observations under consideration are not the direct outputs of a sensor but algorithmically processed versions of these. For example, a robot may obtain images of its surroundings and use algorithms to classify these into walls, doors and hallways, or estimate distances in each direction that it can freely move before encountering an obstacle. It is reasonable to think of situations when these algorithms involve components that can learn during the operation of the agent, alongside the learning of the policy. In such cases, the probabilities of different observations given a state change with time.

The effect of such changes can be demonstrated in a domain as simple as the tiger problem from Cassandra *et al.* [4]. In this problem, there are two doors. Behind one is a tiger and behind the other is some treasure. The world can be in one of two states `tiger-left` and `tiger-right`. An agent can perform one of three actions: `listen`, `open-left` and `open-right`. Listening can provide a noisy observation about which door the tiger is behind. With probability 0.85, it indicates the true door hiding the tiger. There is also a cost of -1 incurred by listening. Opening the door with the tiger results in a reward of -100 and opening the door with the treasure results in a reward of +10.

Clearly the optimal solution in the above problem depends on the exact probability of a `listen` action indicating the true door hiding the tiger. Suppose the agent's sound input mechanism gets damaged and this probability drops to 0.6, the agent must correspondingly reduce the value of listening. In this scenario, if the agent was not aware of the damage, the only way to discover this would be further experience. However, if the agent is provided some indication of the reliability of the sound mechanism, it might be desirable to incorporate knowledge from this into the POMDP.

The ability to account for such sensor variability can be useful in many real-world applications. For example, in a dialog system, reinforcement learning is used to identify how to respond to the user in order to satisfy their request as quickly as possible. Such a system has a component that understands natural language spoken/typed by the user and converts this into a list of hypotheses that form the observation provided to the reinforcement learning agent. If the natural language understanding component is improved, the observations are more reliable and the system can make do with fewer clarifications. Another example is when robots need to identify specific types of objects. If the robot's classifier based on the visual appearance of the object improves, it needs to perform fewer additional tests to confirm the identity of the object. In general, accounting for such variability will be useful in settings consisting of multiple learned modules that interact with each other. It is also useful in settings where physical sensors can fail but the agent has a mechanism to detect that a sensor has failed. For instance, when a robot is sent to Mars, we would want it to perform as best as it can given the limitation of a broken sensor rather than shut down operation completely, or behave exactly as it would have if the sensor had been working.

The standard formulation of POMDPs assumes a fixed observation function. Further, many current solutions to POMDPs assume that this observation function, as well as the transition function, are known [15]. The goal of this project is to extend a mechanism for learning the transition and observation function - Bayes Adaptive POMDPs [15] (outlined in section 3.2) to allow for the case where the observation function is a moving target. The extensions compared are some heuristic modifications to the BAPOMDP belief update to incorporate recency weighting, and a method that attempts to learn a modification to the BAPOMDP belief update that is better suited to tracking a varying observation function.

2 Related Work

Partially Observable Markov Decision Processes (POMDPs) were first introduced in Artifical Intelligence By Kaelbling *et al.* [9] to model stochastic domains in which traditional AI planning methods could not be used. The use of POMDPs allows for the incorporation of uncertainty in both the effects of an agent's actions and its perceptual abilities. This is useful in a number of applications including many robotics tasks where perception is done via noisy sensors [7, 8], goal-directed dialog with an automated system [24], automated fault recovery [10] and medical diagnosis [6].

Assuming the transition, observation and reward model are known, there exist methods such as the Witness algorithm [9] that can solve for the optimal policy, if the state and actions spaces are small and discrete. However, such exact methods quickly become intractable for problems with more than a few tens of states, actions and observations for a number of reasons. Dynamic programming for POMDPs requires updating belief states, a process whose complexity grows exponentially in the number of states. Also, dynamic programming operates over hyperplanes in the space of possible beliefs, which grows exponentially in the number of possible observations [1]. Further, for most applications, as the observation, transition and reward models are not known, exact methods can in any case not be applied.

The problem of control in a POMDP can be broken into two parts [9]. The first is belief or state estimation - which is responsible for updating the belief state based on the last action, current observation and previous belief state. The second is learning a policy given the belief state. Approximate solutions to POMDPs involve approximating the belief state, the value function given the belief state, or both.

Two classes of approaches for tractable belief estimation are the use of factored Bayesian approaches and particle filtering. Factored approaches use domain knowledge to design a state representation and a functional form for the observation function using appropriate independence assumptions [23, 20]. It is then typically possible to maintain beliefs for all possible states based on observations. However, such solutions are typically domain specific and require a fair amount of domain expertise to design appropriate representations. A more general alternative, that has recently been used in the development of fairly scalable algorithms, is particle filtering [18, 21]. Particle filters use observed samples called particles, weighted by their likelihood of being generated to estimate the posterior of unobserved variables given the evidence [21].

Most of the above methods assume that the transition and observation model of the POMDP are known [15]. One approach that does not make this assumption is that of Silver *et al.* [18] that uses a forward simulator to make Monta Carlo estimates of the belief. A more general framework is that of Ross *et al.* [15], which uses Bayesian ideas to maintain estimates of both the model and the actual belief state. The maintained model can be updated from purely online experience. This project attempts to modify this framework to allow for a varying observation function.

Given either an exact or approximate belief state, one class of approaches treat the problem as a regular continuous MDP in the space of belief states. This could be done by either assuming that the agent is in the most probable state, using voting heuristics that combine action values of multiple states weighted by the likelihood of being in that state [1], or using function approximation with a combination of probability values from the belief state and other features [5]. In applications such as dialogue, this approach has met with considerable success, especially with the use of reinforcement learning algorithms with low sample complexity - a property highly desirable in this domain [5, 12].

Another class of solutions is to use policy search methods. As these do not use the assumptions that differentiate between MDPs and POMDPs, they are equally applicable for POMDPs. These include standard policy gradient methods such as REINFORCE [22], an extension of the same to infinite horizon POMDPs [3], or hybrid value-policy approahes such as VAPS [2] which has a tunable parameter that causes the algorithm to vary from Q-learning at one end to REINFORCE on the other. At intermediate values, it is a hybrid algorithm that provides convergence guarantees even for POMDPs. A more scalable variant from this class of methods is PEGASUS - a policy search method that transforms the POMDP to an equivalent POMDP with deteministic transitions and then performs policy search using Monte Carlo estimates of returns, which was successful on a simulated bicycle riding task [11]. The main disadvantages of these methods however, is that they suffer from high variance and it is difficult to obtain enough samples of an observation to get unbiased gradient estimates.

Recently, solvers capable of handling domains with thousands of states have been developed, due to an approach called point based value iteration (PBVI) [17]. The basic idea in this approach is to collect a set of reachable belief points and approximate the value function using a set of vector optimal over these belief points. There are many possible strategies for collecting belief points, ranging from random forward simulation [19] to identifying belief points for which the current approximation is poorest [14]. Using these belief points, value-iteration style backups are performed to update the current value function estimate. These can range from full backups [13], or more efficient iterative backups [19]. An empirical comparison of different variants on a number of standard POMDP domains can be found in Shani *et al.* [17], which indicates that different strategies are likely to be successful in different types of domains.

This work is complementary to the the above approaches in that in principle, it can be combined with any policy learning approach that uses belief states, and allows for sufficient exploration.

3 Background and notation

3.1 Partially Observable Markov Decision Processes

A Partially Observable Markov Decision Process (POMDP) is a tuple $(\mathbb{S}, \mathbb{A}, \mathbb{T}, \mathbb{R}, \mathbb{O}, \mathbb{Z}, \gamma, b_0)$, where $\mathbb{S}$ is a set of states, $\mathbb{A}$ is a set of actions, $\mathbb{T}$ is a transition function, $\mathbb{R}$ is a reward function, $\mathbb{O}$ is a set of observations, $\mathbb{Z}$ is an observation function, γ is a discount factor and b_0 is an initial belief state [9]. These are defined as follows.

At any instant of time t, the agent is in a state $s_t \in \mathbb{S}$. This state is hidden from the agent and only a noisy observation $o_t \in \mathbb{O}$ of s_t is available to it. The agent maintains a belief state b_t which is a distribution over all possible states it could be in at time t. $b_t(s_i)$ gives the probability of being in state s_i at time t. Based on b_t, the agent chooses to take an action $a_t \in \mathbb{A}$ according to a policy π, commonly represented as a probability distribution over actions where $\pi(a_t|b_t)$ is the probability of taking action a_t when the agent is in belief state b_t. On taking action a_t, the agent is given a real-valued reward r_t, transitions to a state s_{t+1}, and receives a noisy observation o_{t+1} of s_{t+1}.

State transitions occur according to the probability distribution $P(s_{t+1}|s_t, a_t) = \mathbb{T}(s_t, a_t, s_{t+1})$, observations are related to the states by the probability distribution $P(o_t|s_t, a_{t-1}) = \mathbb{Z}(o_t, s_t, a_{t-1})$ and rewards obtained follow the distribution $P(r_t|s_t, a_t) = \mathbb{R}(s_t, a_t, s_{t+1})$.

The objective is to identify a policy π that is optimal in the sense that it maximizes the expected long term discounted reward, called return, given by

$$g = \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^t r_t \right]$$

3.2 BAPOMDPs

Bayes Adaptive POMDPs or BAPOMDPs is a model that transforms a POMDP into a form that allows searching for an optimal policy, accounting for both state and parameter uncertainty [15]. Given a POMDP, $(\mathbb{S}, \mathbb{A}, \mathbb{T}, \mathbb{R}, \mathbb{O}, \mathbb{Z}, \gamma, b_0)$, they assume that the state space $\mathbb{S}$, action space $\mathbb{A}$ and observation space $\mathbb{O}$ are finite and known. The transition and observation functions are unknown or partially known,

and represented by experience counts. Let $\phi^a_{ss'}$ be the number of times the transition (s, a, s') is observed and let $\psi^a_{s'z}$ be the number of times action a results in a transition to state s' with observation z. Then the transition and observation functions are given by

$$\begin{aligned} \mathbb{T}^{sas'}_{\phi} &= \frac{\phi^a_{ss'}}{\sum_{s" \in \mathbb{S}} \phi^a_{ss"}} \\ \mathbb{O}^{s'az}_{\psi} &= \frac{\psi^a_{s'z}}{\sum_{z \in \mathbb{Z}} \psi^a_{s'z}} \end{aligned}$$

Consider an agent in state s with count vectors ϕ and ψ. Suppose it takes action a, moves to state s' and observes z. Then ϕ and ψ are respectively updated to $\phi' = \phi + \delta^a_{ss'}$ and $\psi' = \psi + \delta^a_{s'z}$ where $\delta^a_{ss'}$ is a vector of zeroes with a 1 corresponding to $\phi^a_{ss'}$, and $\delta^a_{s'z}$ is a vector of zeroes with a 1 corresponding to $\psi^a_{s'z}$.

The BAPOMDP operates over the state space $\mathbb{S} \times \Phi \times \Psi$. Its transitions and observations are given by

$$\begin{aligned} \mathcal{T}(\langle s, \phi, \psi \rangle, a, \langle s', \phi', \psi' \rangle) &= \begin{cases} \mathbb{T}^{sas'}_{\phi} \mathbb{O}^{s'az}_{\psi} & , \phi' = \phi + \delta^a_{ss'}, \psi' = \psi + \delta^a_{s'z} \\ 0 & , otherwise \end{cases} \\ \mathcal{O}(\langle s, \phi, \psi \rangle, a, \langle s', \phi', \psi' \rangle, z) &= \begin{cases} 1 & , \phi' = \phi + \delta^a_{ss'}, \psi' = \psi + \delta^a_{s'z} \\ 0 & , otherwise \end{cases} \end{aligned}$$

It has the same reward function $\mathbb{R}$ and discount factor γ as the original POMDP, and the initial belief is given by

$$b'_0(s, \phi_0, \psi_0) = \begin{cases} b_0(s) & , \phi = \phi_0, \psi = \psi_0 \\ 0 & , otherwise \end{cases}$$

To make this a finite POMDP, given a parameter ϵ, it is possible to limit the range of possible Φ and Ψ, and still obtain an ϵ-approximation of the value function [15]. The conditions in the transition and observation function are then modified with an appropriate projection operation.

With this approximation, the BAPOMDP is a discrete POMDP with a known model, and if observation z is obtained after performing action a at time t, the belief can be exactly updated as follows -

$S_{NZ} = \{(s, \phi, \psi) \in \mathbb{S} \times \Phi \times \Psi : b'_t(s, \phi, \psi) > 0\}$

`for` $(s, \phi, \psi) \in S_{NZ}$:

    `for` $s' \in \mathbb{S}$:

        $\phi' = \phi + \delta^a_{ss'}$

        $\psi' = \psi + \delta^a_{s'z}$

        $b'_{t+1}(s', \phi', \psi') = 0$

`for` $(s, \phi, \psi) \in S_{NZ}$:

    `for` $s' \in \mathbb{S}$:

        $\phi' = \phi + \delta^a_{ss'}$

        $\psi' = \psi + \delta^a_{s'z}$

        $b'_{t+1}(s', \phi', \psi') += b'_t(s, \phi, \psi) * \mathbb{T}^{sas'}_{\phi} * \mathbb{O}^{s'az}_{\psi}$

However, this allows the vectors ϕ and ψ to grow indefinitely, resulting in an infinite state space. Even for very simple problems such as the Tiger problem [4], exact belief updates becomes intractable in a very small number of iterations. However, various approximations are proposed in Ross *et. al.*, 2007 [15] to make these updates tractable. The approximation adopted for the purposes of this project is to retain the top-n beliefs after each update. Ross *et. al.*, 2007 [15] demonstrate that this strategy works fairly well, outperforming more sophisticated approaches such as Monte Carlo approximation, and is fairly insensitive to the value of n.

Now both belief estimation, and estimation of a transition and reward function can be performed. Different planning techniques can then be used to identify an appropriate policy.

4 Formal Problem Definition

Given a POMDP $(\mathbb{S}, \mathbb{A}, \mathbb{T}, \mathbb{R}, \mathbb{O}, \mathbb{Z}, \gamma, b_0)$ where $\mathbb{Z}$ varies with time as $\mathbb{Z}_t$, and given an auxilliary measure f_t such that there exists a relation $\mathbb{Z}_t = \omega(f_t)\ \forall\ t$, how can solutions to POMDPs be modified to make use of this information? Ideally, such solutions should not require the knowledge of the relation ω. However, in several use cases, even a solution that requires the knowledge of ω is useful.

5 Proposed Solutions

This project attempts to extend the BAPOMDP formulation to the case when the observation function being estimated is a moving target.

5.1 Larger magnitude pseudo-count updates

The simplest variant is to replace $\delta^a_{s'z}$ with some $c > 1$ instead of 1 for $\psi^a_{s'z}$. This increases the magnitude of each belief update. Since after each belief update, only the top-n beliefs are retained, followed by normalization, this is expected perform some form of recency weighting. This method is referred to in the experiments as `large_updates`.

5.2 Normalization of observation counts

An alternative mechanism to perform recency weighting is to normalize all current observation count vectors ψ having a non-zero belief, when the value of the indicator f_t changes. Normalization of a vector ψ does not change the observation function estimate it corresponds to, but causes more recent updates to have a larger impact as the magnitude of the increment $\delta^a_{s'z}$ becomes large compared to value of different components in the vector ψ. This method is referred to in the experiments as `normalize_all`.

5.3 Conditional normalization of observation counts

Normalization of ψ on every change in f_t can result in high variance because the magnitude of the increment $\delta^a_{s'z}$ becomes large compared to value of different components in the vector ψ. A less extreme form of recency weighting is to track the value of f_t when the vectors ψ were last normalized and re-normalize them when the current value of f_t differs from this by more than a threshold value. This is likely to reduce variance as larger updates occur in fewer iterations. This method is referred to in the experiments as `normalize_threshold`.

5.4 Learning the modification to observation counts

One of the problems with the method `normalize_all` is that it decreases the magnitude of each plausible ψ vector to 1 every time the observation function changes. If this is very frequent, the updates may be too large, and overshoot the target every time, similar to the effect of a very large learning rate in stochastic gradient descent. However, it may still be possible to adjust the magnitude of the vector ψ so that new updates are upweighted but not by so much that a continuously varying observation function cannot be tracked. Also, it is possible that different schedules of variation of the observation function would require different hyperparameters in the previously proposed methods. Instead, it would be more desirable to learn how to modify each ψ vector currently maintained in the belief based on the current and previous values of the observation indicator, that is, f_t and f_{t-1}.

In the default BAPOMDP belief update, on observing observation z after performing action a, the belief of observation count ψ is used to update beliefs of all count vectors -

$$\psi' = \psi + \delta^a_{s'z}\ \ \forall\ s' \in \mathbb{S}$$

The proposed method `normalize_all` modifies this, when $|f_t - f_{t-1}| > 0$, to using the belief of ψ to update the belief of -

$$\psi' = \frac{\psi}{\|\psi\|} + \delta^a_{s'z}\ \ \forall\ s' \in \mathbb{S}$$

This can be generalized to using the belief of ψ to update the belief of all

$$\psi' = c\psi + \delta^a_{s'z} \;\; \forall \; s' \in \mathbb{S} \tag{1}$$

where c is a scalar that depends on f_t and f_{t-1}.

A neural network with a single hidden layer is used to model the mapping between f_t, f_{t-1} and c as follows -

$$\begin{aligned} x &= [f_t \;\; f_{t-1} \;\; 1]^T \\ h &= \sigma(\alpha x), \;\; \alpha \in \mathbb{R}^{H \times 3} \\ c &= \beta^T h + \gamma, \;\; \beta \in \mathbb{R}^H, \gamma \in \mathbb{R} \end{aligned}$$

where H is the number of units in the hidden layer, and σ is the sigmoid function.

Here, α, β and γ are learnable parameters. However, in order to train the network, a loss function is needed. This requires either target values of c or a relation between the observation counts ψ and the indicators f_t. It is not clear how to directly obtain targets for c besides providing the hyperparameter values tuned for the heuristics. However, using this as a target will, in the best case, only provide an approximation of the heuristic, which is not the desired goal. Hence, we choose to assume a function relation between the desired observation count vector ψ^* and f_t. That is, we assume that for some known function g, when $f_t = g(\psi^*)$, the observation function induced by ψ^* is a good approximation of the true observation function. Then, if the current estimated observation count vector ψ is modified to ψ' as in equation 1 using a value of c predicted using f_t and f_{t-1}, we can use the squared error between $g(\psi')$ and the indicator at the next step f_{t+1} as a loss to train the network.

A challenge with this idea is that we do not have a single value of ψ and ψ' at each time step but a belief over different possible values of ψ and ψ'. For simplicity, we consider that ψ' with maximum marginal probability according to the BAPOMDP belief state after the belief update at time step t. This would have been obtained from some ψ as $\psi' = c\psi + \delta^a_{s'z}$ for some s', where z is the observation obtained at time t and a is the action taken that resulted in it. A constraint then placed on the choice of the function g is that it must be possible to compute the derivative of $g(\psi')$ with respect to c. Let $g'(\psi')$ represent this derivative. Using this choice of ψ and ψ', we have,

$$\begin{aligned} L &= (g(\psi') - f_{t+1})^2 \\ \Rightarrow \frac{\partial L}{\partial c} &= 2(g(\psi') - f_{t+1})g'(\psi') \end{aligned}$$

Also,

$$\begin{aligned} c &= \beta^T h + \gamma \\ &= \sum_k \beta_k h_k + \gamma \\ \Rightarrow \frac{\partial c}{\partial \beta_k} &= h_k \\ \frac{\partial c}{\partial h_k} &= \beta_k \\ \frac{\partial c}{\partial \gamma} &= 1 \end{aligned}$$

The gradient of the loss with respect to γ is given by

$$\frac{\partial L}{\partial \gamma} = \frac{\partial L}{\partial c}\frac{\partial c}{\partial \gamma} = 2(g(\psi') - f_{t+1})g'(\psi') \tag{2}$$

The gradient of the loss with respect to β_k is given by

$$\frac{\partial L}{\partial \beta_k} = \frac{\partial L}{\partial c}\frac{\partial c}{\partial \beta_k} = 2(g(\psi') - f_{t+1})g'(\psi')h_k$$

This can also be written as a single vector as follows -

$$\frac{\partial L}{\partial \beta} = 2(g(\psi') - f_{t+1})g'(\psi')h \tag{3}$$

Also,

$$\begin{aligned} h &= \sigma(\alpha x) \Rightarrow h_i = \sigma\left(\sum_j \alpha_{ij} x_j\right) \\ \Rightarrow \frac{\partial h_k}{\partial \alpha_{ij}} &= \sigma'\left(\sum_j \alpha_{kj} x_j\right) \frac{\partial \sum_j \alpha_{kj} x_j}{\partial \alpha_{ij}} \\ &= \begin{cases} 0, & i \neq k \\ \sigma'\left(\sum_j \alpha_{kj} x_j\right) x_j, & i = k \end{cases} \end{aligned}$$

But since $\sigma(x)$ is the sigmoid function

$$\begin{aligned} \sigma'(x) &= \sigma(x)(1 - \sigma(x)) \\ \Rightarrow \sigma'\left(\sum_j \alpha_{kj} x_j\right) &= \sigma\left(\sum_j \alpha_{kj} x_j\right)\left(1 - \sigma\left(\sum_j \alpha_{kj} x_j\right)\right) \\ &= h_k(1 - h_k) \\ \Rightarrow \frac{\partial h_k}{\partial \alpha_{ij}} &= \begin{cases} 0, & i \neq k \\ h_k(1 - h_k)x_j = h_i(1 - h_i)x_j, & i = k \end{cases} \end{aligned} \tag{4}$$

Then,

$$\begin{aligned} \frac{\partial c}{\partial \alpha_{ij}} &= \frac{\partial \sum_k \beta_k h_k + \gamma}{\partial \alpha_{ij}} \\ &= \sum_k \beta_k \frac{\partial h_k}{\partial \alpha_{ij}} \\ &= \beta_i h_i(1 - h_i)x_j \quad \text{(using equation 4)} \end{aligned}$$

This gives

$$\begin{aligned} \frac{\partial L}{\partial \alpha_{ij}} &= \frac{\partial L}{\partial c}\frac{\partial c}{\partial \alpha_{ij}} \\ &= 2(g(\psi') - f_{t+1})g'(\psi')\beta_i h_i(1 - h_i)x_j \end{aligned}$$

This can be written in matrix form as

$$\frac{\partial L}{\partial \alpha} = 2(g(\psi') - f_{t+1})g'(\psi')(\beta \odot h \odot (1 - h))x^T \tag{5}$$

where $\odot$ represents elementwise multiplication (Hadamard product).

Using equations 5, 3 and 2, we can get the stochastic gradient descent update rules with learning rate η

$$\alpha \leftarrow \alpha - \eta(g(\psi') - f_{t+1})g'(\psi')(\beta \odot h \odot (1 - h))x^T \tag{6}$$

$$\beta \leftarrow \beta - \eta(g(\psi') - f_{t+1})g'(\psi')h \tag{7}$$

$$\gamma \leftarrow \gamma - \eta(g(\psi') - f_{t+1})g'(\psi') \tag{8}$$

This allows the network to be updated given f_{t+1}. This method is referred to later as `learned_updates`.

6 Experimental Setup

The proposed solutions have been compared on the modified tiger domain discussed in section 1. This is characterized as follows -

States = { `tiger-left`, `tiger-right` }

Actions = { `listen`, `open-left`, `open-right` }

Observations = { `tiger-left`, `tiger-right` }

The initial belief is uniform. All transitions are identity. The reward is -100 for opening the door on a tiger, +10 for opening the other door, and -1 for every turn of listening. The episode terminates when a door is opened.

On listening, the observation is equal to the true state with probability $p(t)$. Experiments have been performed on multiple possible functions $p(t)$. This value $p(t)$ is directly provided as the indicator f_t.

In order to separate errors in observation function estimation, and estimation of other parameters, the true transition and reward models are provided to the agent, and kept fixed. The agent is only expected to perform belief estimation of its state, and estimate observation probabilities for the `listen` action. A similar setting was used in Ross *et. al.*, 2007 [15] to demonstrate the ability of BAPOMDPs to estimate the observation function.

All experiments use a hand-coded policy of K `listen`s followed by opening the door less likely to be hiding the tiger, as indicated by the belief state. This ensures that all methods compared obtain the same number of observation samples for the `listen` action, and also avoids the problem of ensuring that the policy maintains a good balance between exploration and exploitation. The methods proposed in this project are all modifications to the belief monitoring step, and in principle, it should be possible to combine any planning method that allows for sufficient exploration, with them.

For each method, at each time step, an estimate $\hat{p}(t)$ is obtained for $p(t)$, as follows -

$$\hat{p}(t) = \frac{\mathbb{O}_{\hat{\psi}}^{s_1 a z_1} + \mathbb{O}_{\hat{\psi}}^{s_2 a z_2}}{2}$$

where $a =$ `listen`, $s_1 = z_1 =$ `tiger-left`, $s_2 = z_2 =$ `tiger-right` and $\hat{\psi}$ is the vector ψ with maximum marginal probability according to the belief state at time t, that is,

$$\hat{\psi} = \max_{\psi} \sum_{s',\phi} b'_t(s', \phi, \psi)$$

The methods are then be compared by plotting $p(t)$ and $\hat{p}(t)$ with time, and using the mean squared error between the two.

The `learned_updates` method also requires an appropriate choice of the function g indicating a relation between the desired observation count vector ψ^* and f_t. We choose the following function

$$g(\psi^*) = \frac{\psi^{*a_0}_{s_1 z_1} + \psi^{*a_0}_{s_2 z_2}}{2}$$

where $a_0 =$ `listen`, $s_1 = z_1 =$ `tiger-left`, $s_2 = z_2 =$ `tiger-right`. This is chosen because it allows for a simple form of $g'(\psi')$ derived as follows -

$$\begin{aligned}
\psi' &= c\psi + \delta^a_{s'z} \text{ for some } a,\ s',\ z \\
\Rightarrow g'(\psi') &= g(c\psi + \delta^a_{s'z}) \\
&= \frac{\partial}{\partial c}\left(\frac{c\psi^{a_0}_{s_1z_1} + \delta^a_{s'z}[(s_1,a_0,z_1)] + c\psi^{a_0}_{s_2z_2} + \delta^a_{s'z}[(s_2,a_0,z_2)]}{2}\right) \\
&\quad \text{with some abuse of notation wherein} \\
&\quad \begin{cases} \psi^{a_0}_{s_1z_1} \text{ represents the component of vector } \psi \text{ corresponding to the tuple } (s_1,a_0,z_1) \\ \delta^a_{s'z}[(s_1,a_0,z_1)] \text{ represents the component of vector } \delta^a_{s'z} \text{ corresponding to the tuple } (s_1,a_0,z_1) \\ \psi^{a_0}_{s_2z_2} \text{ represents the component of vector } \psi \text{ corresponding to the tuple } (s_2,a_0,z_2) \\ \delta^a_{s'z}[(s_2,a_0,z_2)] \text{ represents the component of vector } \delta^a_{s'z} \text{ corresponding to the tuple } (s_2,a_0,z_2) \end{cases} \\
&= \frac{\psi^{a_0}_{s_1z_1} + \psi^{a_0}_{s_2z_2}}{2} = g(\psi)
\end{aligned}$$

This can be substituted in equations 6, 7 and 8 to get the following update rules -

$$\begin{aligned}
\alpha &\leftarrow \alpha - \eta(g(\psi') - f_{t+1})g(\psi)(\beta \odot h \odot (1-h))x^T \\
\beta &\leftarrow \beta - \eta(g(\psi') - f_{t+1})g(\psi)h \\
\gamma &\leftarrow \gamma - \eta(g(\psi') - f_{t+1})g(\psi)
\end{aligned}$$

α is initialized to random values sampled uniformly from $[0,1]$. β is initialized to a zero vector and γ to 1 so that the network initially mimics the default BAPOMDP update.

Experiments use $n = 10$ beliefs retained after each update for all methods, and $K = 10$ `listen`s in the policy. The performance was found to be relatively insensitive to both n and K. However, increasing n and K would increase the average return simply because more observations have been taken and hence the belief is more likely to indicate the true position of the tiger.

Two possible schedules of variation for $p(t)$ are considered. The first is a linear schedule -

$$p(t) = 0.625 * 1.0003^t$$

and the second is a stepwise schedule -

$$p(t) = 0.625 * 1.03^{[t/100]}$$

where $[\cdot]$ denotes the greatest integer function.

7 Results and Discussion

The `learned_update` method was found not to converge in 1000 or even 10000 episodes for a wide range of hyperparamters: number of hidden units $H \in \{1,2,5,10,20,50,100\}$ and learning rates $\eta \in \{10^-2, 10^-3, 10^-4, 10^-5, 10^-6\}$ for either schedule. Figure 1 shows the performance in a sample run with $H = 10$ and $\eta = 10^{-3}$. It is not clear whether this is because of a poor choice of the function g or due to the use of the maximum likelihood estimate of ψ'.

Another general trend observed was that `large_updates` typically performs about the same as the default BAPOMDP update, regardless of the choice of c. This is likely an indicator that using larger updates and dropping all but the top beliefs does not result in much recency weighting.

Figure 2 shows the performance of the various heuristic methods averaged across 10 trials. It is observed that the `normalize_threshold` method outperforms the default BAPOMDP update, consistently obtaining very low mean squared error. This demonstrates that normalization is a more effective form of recency weighting that is capable of tracking at least some types of changes in the observation function. It is likely that `normalize_all` is unable to perform as well because normalization occurs so frequently that updates are given too much weight which worsens the overall approximation, similar to the effect of a very large learning rate in stochastic gradient descent.

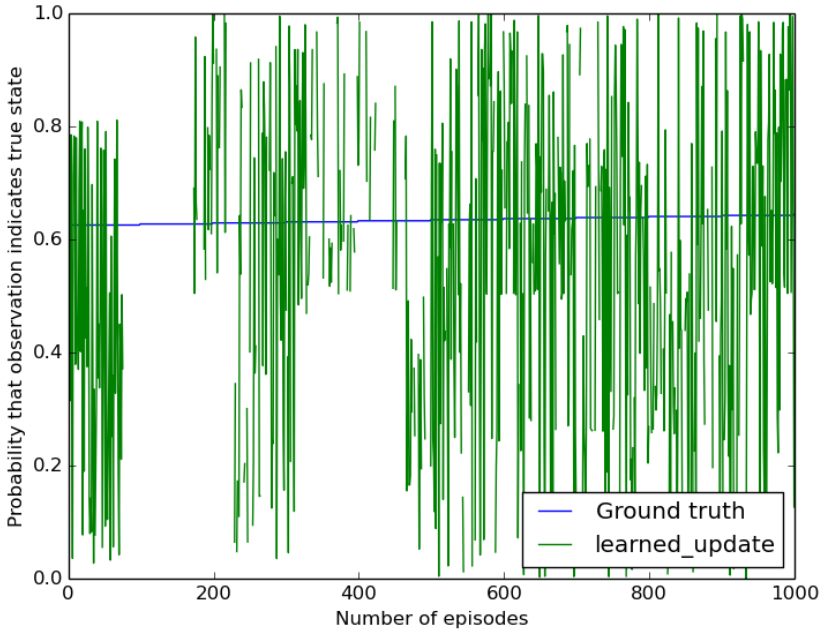

Figure 1: Sample performance `learned_update` with $H = 10$, $\eta = 10^{-3}$ on stepwise variation of $p(t)$ with $K = 10$

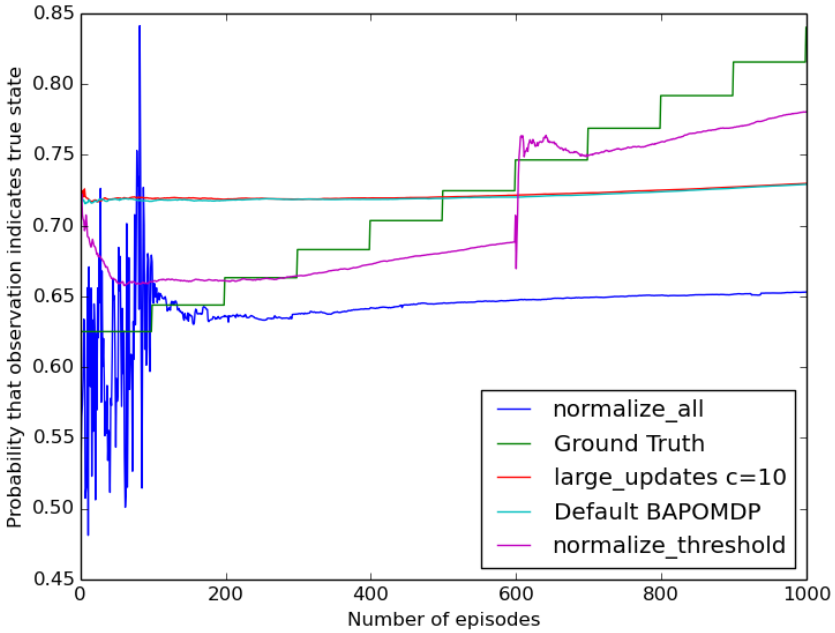

Figure 2: Average performance of heuristic methods with stepwise variation of $p(t)$ and $K = 10$ over 10 trials

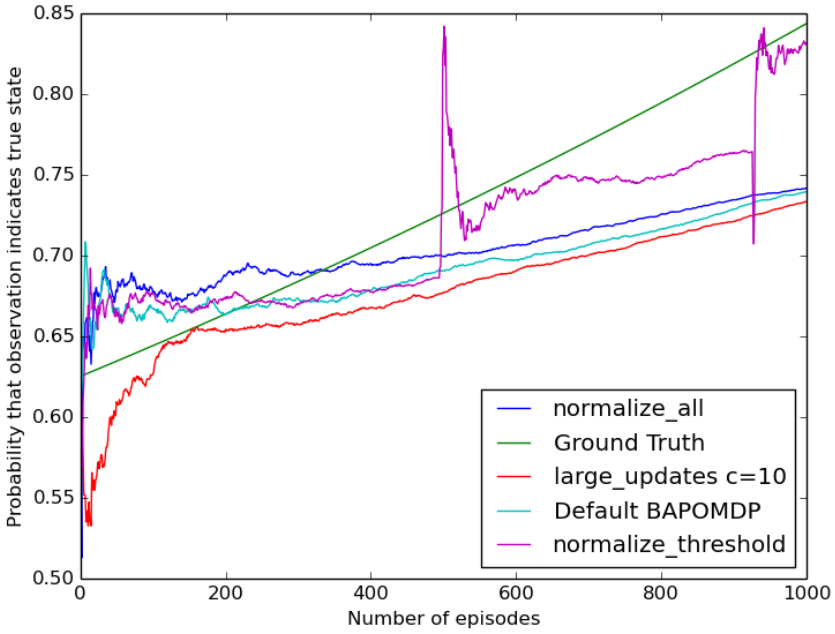

Figure 3: Sample performance of heuristic methods with linear variation of $p(t)$ and $K = 10$

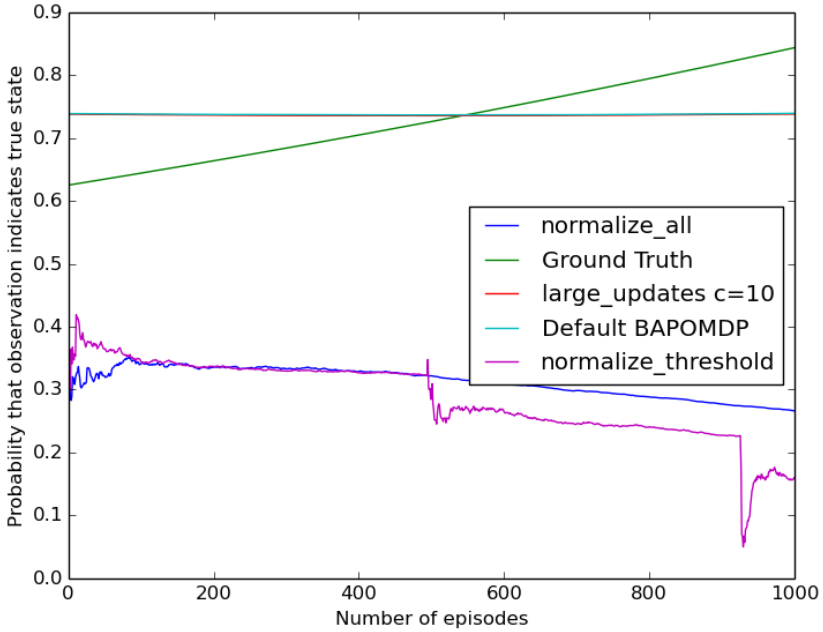

Figure 4: Sample performance of heuristic methods with linear variation of $p(t)$ and $K = 10$

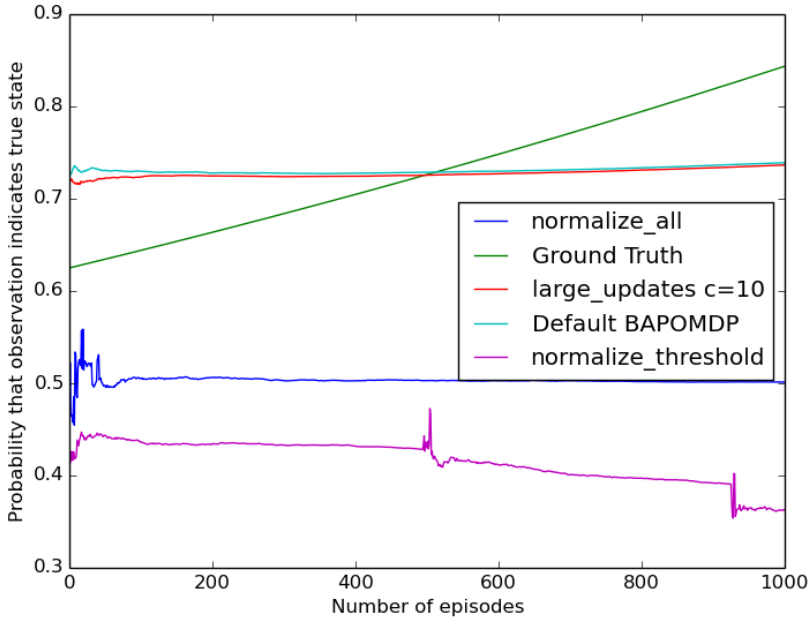

Figure 5: Average performance of heuristic methods with linear variation of $p(t)$ and $K = 10$ over 10 trials

However, with the linear schedule, there is much more variability in the performance of both the `normalize_all` and `normalize_threshold` methods. On some runs such as that shown in figure 3, these methods perform fairly well, with `normalize_threshold` obtaining a very low overall mean squared error of about 0.005. In this run, both `normalize_all` and `normalize_threshold` outperform the default BAPOMDP update, and the `large_updates` method, whose performance is nearly identical to the default BAPOMDP update. However, on other runs such as that shown in figure 4, they perform much worse than the default BAPOMDP update. While variance is expected in these methods, it is more expected across episodes rather than across trials. Hence it is not clear why the overall performance is very poor in some trials. Such trials are in fact more frequent than the successful trials, causing an average over 10 trials to look like figure 5.

It is important to note that in all these experiments, in the initial belief state, the only observation count vector ψ with non-zero probability is indicative of the true initial observation function. Further, in the poor runs, a larger number of episodes is not sufficient to allow the methods to recover unless the observation function saturates at 1.0. The poor performance on the linear schedule indicates that some recency weighting heursitics can perform well for some changes in the observation function, but they perform poorly if the change is continuous. This is probably because when the observation function is continuously changing, it is not always possible to obtain enough samples to get a good estimate of the observation function before the current estimate is down-weighted by normalization.

8 Conclusion

This project examined a number of heuristic modifications to the BAPOMDP belief update, all of which were aimed at performing recency weighting in various forms, as well as a learned modification to the BAPOMDP belief update to track a varying observation function in a POMDP, given an indicator of the current observation function. The best perfromance was obtained by normalizing the observation count vector when the observation function indicator changed by a significant amount. This method was found to perform better than the default BAPOMDP update when the observation function varied in steps but was not able to handle the case when the observation function changed continuously.

9 Future Work

Further exploration is still required to determine why the `learned_update` method was not able to converge even in a large number of episodes because it is at least capable of modelling the default BAPOMDP update and the `normalize_all` update. It is possible that a recurrent model, that is allowed to store information from all previous values of f_t, may perform better. It is also possible that variants of the `normalize_threshold` method, where the observation count vector is scaled down but not entirely normalized, may be able to handle the case of a continuously changing observation function.

In addition to this, the experiments in this project were performed on a very small domain. It is necessary to test the scalability and performance of the propsed methods on larger domains, which critically depend on the scalability of BAPOMDPs to larger domains. Another important set of experiments would be to examine how different methods proposed here combine with more sophicticated planning techniques such as point based value iteration.

Further, this project extends a belief moitoring algorithm in the BAPOMDP framework that assumes a discrete state, action and observation space. An interesting future direction would be to apply these or similar extensions to other BAPOMDP solution techniques that allow for continuous spaces such as that of Ross *et. al.*, 2008 [16].

References

[1] Douglas Aberdeen. A (revised) survey of approximate methods for solving partially observable markov decision processes. *National ICT Australia, Canberra, Australia*, 2003.

[2] Leemon Baird and Andrew W Moore. Gradient descent for general reinforcement learning. *Advances in neural information processing systems*, pages 968–974, 1999.

[3] Jonathan Baxter, Peter L Bartlett, et al. Reinforcement learning in pomdp's via direct gradient ascent. In *ICML*, pages 41–48. Citeseer, 2000.

[4] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, August 1994.

[5] M Gašić, F Jurčíček, Simon Keizer, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. Gaussian processes for fast policy optimisation of pomdp-based dialogue managers. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 201–204. Association for Computational Linguistics, 2010.

[6] Milos Hauskrecht and Hamish Fraser. Planning treatment of ischemic heart disease with partially observable markov decision processes. *Artificial Intelligence in Medicine*, 18(3):221–244, 2000.

[7] Kaijen Hsiao, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Grasping pomdps. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4685–4692. IEEE, 2007.

[8] Vu Anh Huynh and Nicholas Roy. iclqg: combining local and global optimization for control in information space. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2851–2858. IEEE, 2009.

[9] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial intelligence*, 1998.

[10] Michael L Littman, Nishkam Ravi, Eitan Fenson, and Rich Howard. An instance-based state representation for network repair. In *AAAI*, pages 287–292, 2004.

[11] Andrew Y Ng and Michael Jordan. Pegasus: A policy search method for large mdps and pomdps. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 406–415. Morgan Kaufmann Publishers Inc., 2000.

[12] Olivier Pietquin, Matthieu Geist, and Senthilkumar Chandramohan. Sample efficient on-line learning of optimal dialogue policies with kalman temporal differences. In *IJCAI 2011*, pages 1878–1883, 2011.

[13] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, pages 1025–1032, 2003.

[14] Joelle Pineau and Geoffrey J Gordon. Pomdp planning for robust robot control. In *Robotics Research*, pages 69–82. Springer, 2007.

[15] Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-adaptive pomdps. In *Advances in neural information processing systems*, pages 1225–1232, 2007.

[16] Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayesian reinforcement learning in continuous pomdps with application to robot navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2845–2851. IEEE, 2008.

[17] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based pomdp solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

[18] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[19] Matthijs TJ Spaan and Nikos Vlassis. Perseus: Randomized point-based value iteration for pomdps. *Journal of artificial intelligence research*, 24:195–220, 2005.

[20] Blaise Thomson, Jost Schatzmann, and Steve Young. Bayesian update of dialogue state for robust dialogue systems. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4937–4940. IEEE, 2008.

[21] Jason D Williams. Using particle filters to track dialogue state. In *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, pages 502–507. IEEE, 2007.

[22] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[23] Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174, 2010.

[24] Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.