Learning in non-stationary Partially Observable Markov Decision Processes

Robin JAULMES, Joelle PINEAU, Doina PRECUP

McGill University, School of Computer Science, 3480 University St., Montreal, QC, H3A2A7

Abstract. We study the problem of finding an optimal policy for a Partially Observable Markov Decision Process (POMDP) when the model is not perfectly known and may change over time. To help us in this learning task we assume the availability of an oracle, that can provide information about the underlying state. We present the algorithm MEDUSA+, which incrementally improves a POMDP model using oracle queries, while still optimizing the reward. Empirical results show the response of the algorithm to changes in the parameters of a model: the changes are learned quickly and the agent still accumulates high reward throughout the process.

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) are a well-studied framework for sequential decision-making in partially observable domains[7]. Many recent algorithms have been proposed for doing efficient planning in POMDPs [9];[10];[13]. However most of these rely crucially on having a known and stable model of the environment. On the other hand, the experience-based approaches also need a stationary model [8];[2];[12]. Furthermore, they require very large amounts of data, which would be hard to obtain in a realistic setup.

In many applications it is relatively easy to provide a rough model, but much harder to provide an exact one. Furthermore, because the model may experience changes with time, we would like to use experimentation to improve our initial model. The overall goal of this work is to combine a partial model of the environment with direct experimentation, in order to produce solutions that are robust to model uncertainty and evolution, while scaling to large domains. To do that, we assume that uncertainty is part of the model and design our agent to take it into account when making decisions.

The technique we propose in this paper is an algorithm called MEDUSA+, which is an improvement of the MEDUSA algorithm we presented in [6]. It is based on *active learning*[4], which is a well-known problem formulation in machine learning for classification tasks with sparsely labelled data. In active learning the goal is to select which examples should be labelled by considering the expected information gain. As detailed by Anderson and Moore [1], these ideas extend nicely to dynamical systems such as HMMs.

We will assume in the present work the availability of an oracle that can provide the agent with exact information about the current state, upon request. While the agent experiments with the environment, it can ask for a query in order to obtain state information, when this is deemed necessary. The exact state information is used only to improve the model, not in the action selection process, which means that we may have a delay between the query request and the query processing. This is a realistic assumption, since in a lot of applications (robotics, speech management), it is easy to determine the exact states we went through *after* the experimentation has taken place.

The model uncertainty is represented by a Dirichlet distribution over all possible models, in a method inspired from Dearden et al. [5] and its parameters are updated whenever new experience is acquired. They also decay with time, so recent experience has more weight than old experience: it is a useful feature when the parameters of the POMDP are not stationary.

The paper is structured as follows. In Section 2 we review the basic POMDP framework. Section 3 describes our algorithm MEDUSA+, and outlines modifications that we made compared to MEDUSA. Section 4 shows the theoretical properties of MEDUSA+. Section 5 shows the performance of MEDUSA+ on standard POMDP domains. The discussion is in Section 6.

2 Partially Observable Markov Decision Processes

We assume the standard POMDP formulation (Kaelbling et al., 1998); namely, a POMDP consists of a discrete and finite set of states S, of actions A and of observations Z. It has transition probabilities $\{P^a_{s,s'}\} = \{p(s_{t+1} = s'|s_t = s, s_t = a)\}, \forall s \in S, \forall a \in A, \forall s' \in S$ and observation probabilities $\{O^a_{s,z}\} = \{p(z_t = z|s_t = s, a_{t-1} = a)\}, \forall z \in Z, \forall s \in S, \forall a \in A$. It also has a discount factor $\gamma \in [0;1]$ and a deterministic reward function $R : S \times A \times S \times Z \rightarrow \mathbb{R}$, such that $R(s_t, a_t, s_{t+1}, z_{t+1})$ is the immediate reward for the corresponding transition.

At each time step, the agent is in an unknown state $s_t \in S$. It executes an action $a_t \in A$, arrives in an unknown state $s_{t+1} \in S$ and gets an observation $z_{t+1} \in Z$. Agents using POMDP planning algorithms typically keep track of the belief state $b \in \mathbb{R}^{|S|}$, which is a probability distribution over all states given the history experienced so far. A policy is a function that associates an action to each possible belief state. Solving a POMDP means finding the policy that maximizes the expected discounted return $E(\sum_{t=1}^{T} \gamma^t R(s_t, a_t, s_{t+1}, z_{t+1}))$.

While finding an exact solution to a POMDP is computationally intractable, many methods exist for finding approximate solutions. In this paper, we use a point-based algorithm ([9]), in order to compute POMDP solutions. However, the algorithms that we propose can be used with other approximation methods.

We assume the learner knows the reward function, since it is directly linked to the task that it wants to execute, and we focus on learning $\{P^a_{s,s'}\}$ and $\{O^a_{s,z}\}$. These probability distributions are typically harder to specify correctly by hand, especially in real applications. They may also be changing over time. For instance in robotics, the sensor noise and motion error are often unknown and may also vary with the amount of light, the wetness of the floor, or other parameters that might not be known in advance. To learn the transition and observation models, we assume that our agent has the ability to ask a query that will correctly identify the current state. This is a strong assumption, but not entirely unrealistic, since first of all we will not need to know the query result immediately. In fact, in many tasks it is possible (but very costly) to have access to the full state information; it usually requires asking a human to label the state. As a result, clearly we want the agent to make as few queries as possible.

3 The MEDUSA+ algorithm

In this section we describe the MEDUSA+ algorithm. Its main idea is to represent the model uncertainty with a Dirichlet distribution over possible models, and to update directly the parameters of this distribution as new experience is acquired. It is also built so that it can cope with non-stationary environments, where the parameters evolve with time. Furthermore, MEDUSA+ uses queries more efficiently, through the use of the alternate belief and the non-query learning.

This approach scales nicely: we need one Dirichlet parameter for each uncertain POMDP parameter, but the size of the underlying POMDP representation remains unchanged, which means that the complexity of the planning problem does not increase. However this approach requires the agent to repeatedly sample POMDPs from the Dirichlet distribution and solve them, before deciding on the next query.

3.1 Dirichlet Distributions

Consider a N-dimensional multinomial distribution with parameters $(\theta_1, \dots \theta_N)$. A Dirichlet distribution is a probabilistic distribution over these parameters. The Dirichlet itself is parameterized by hyper-parameters $(\alpha_1, \dots \alpha_N)$. The likelihood of the multinomial parameters is defined by:

$$p(\theta_1 \dots \theta_N | D) = \frac{\prod_{i=1}^N \theta_i^{\alpha_i - 1}}{Z(D)}, \text{ where } Z(D) = \frac{\prod_{i=1}^N \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^N \alpha_i)}$$

The maximum likelihood multinomial parameters $\theta_1^* \dots \theta_N^*$ can be computed, based on this formula, as:

$$\theta_i^* = \frac{\alpha_i}{\sum_{k=1}^N \alpha_k}, \forall i = 1, \dots N$$

The Dirichlet distribution is convenient because its hyper-parameters can be updated directly from data. For example, if instance $X = i$ is encountered, α_i should be increased by 1. Also, we can sample from a Dirichlet distribution conveniently using Gamma distributions.

In the context of POMDPs, model parameters are typically specified according to multinomial distributions. Therefore, we use a Dirichlet distribution to represent the uncertainty over these. We use Dirichlet distributions for each state-action pair where either the transition probabilities or the observation probabilities are uncertain.

Note that instead of using increments of 1 we use a learning rate, λ, which measures the importance we want to give to one query.

3.2 MEDUSA

The name MEDUSA comes from *Markovian Exploration with Decision based on the Use of Sampled models Algorithm*. We present here briefly the algorithm as it was described in [6].

First, our agent samples a number of POMDP models according to the current Dirichlet distribution. The agent then computes the optimal policy for each of these models, and at each time step one of those is chosen with a probability that depends on the weights these models have in the current Dirichlet distribution. This allows us to obtain reasonable performance throughout the active learning process: the quality of the sampled models will be linked to the quality of the actions chosen. This also allows the agent to focus the active learning in regions of the state space most often visited by good policies.

Note that with MEDUSA we need not specify a separate Dirichlet parameter for each unknown POMDP parameter. It is often the case that a small number of hyper-parameters suffice to characterize the model uncertainty. For example, noise in the sensors may be highly correlated over all states and therefore we could use a single set of hyper-parameters for all states. In this setup, the corresponding hyper-parameter would be updated whenever action a is taken and observation z is received, regardless of the state.[1] This results in a very expressive framework to represent model uncertainty. As we showed in our previous work, we can vary the number of hyper-parameters to trade-off the number of queries versus model accuracy and performance.

Each time an action is made and an observation is received, the agent can decide to query the oracle for the true identity of the hidden state [2]. If we do query the Dirichlet distributions are updated accordingly.

In our previous paper we argued and showed experimental evidence that under a stable model and such a near-optimal policy the queries always bring useful information up to a point where the model is well-enough learned. In MEDUSA+ we will however modify this setting so that we know when it is optimal to use the oracle and how we can extract as much information as possible at each query.

3.3 Non-Query Learning

A major improvement in MEDUSA+ is the use of *non-query learning*, which consists of (1) deciding when to query and (2) when we don't query, using the information we have from the action-observation sequence and knowledge extracted from the previous queries to update the Dirichlet parameters.

To do an efficient non-query learning we introduce the concept of an alternate belief β. Actually, for each model we keep track of it in addition to the standard belief. The alternate belief is updated in the same way as the standard one, with each action and observation. The only difference is that when a query is done, it is modified so that the

[1] As a matter of fact, we have a function that maps any possible transition or observation parameters to either a hyper parameter or to a *certain* parameter.

[2] Note that theoretically we need not have the query result immediately after it is asked, since the result of the processing is not mandatory for the decision making phase. However we will suppose in the present work that we do have this result immediately.

state is now certain according to it. This allows to keep track of the information still available from the latest query.

The decision of when to make a query or not can be based on the use of different indicators:

– `PolicyEntropy`: the entropy of the resulting MEDUSA policy Ψ. It is defined by: `Entropy`$= -\sum_{a=1}^{|A|} p(\Psi,a)\ln(p(\Psi,a))$. With experimentation we see that this indicator is biased. The fact that all models agree does not necessarily means that no query is needed.
– `Variance`: the variance over the values that each model computes for its optimal action. It is defined by: `Variance`$= \sum_i (Q(m_i,\Pi(h,m_i)) - \hat{Q})^2$. Experiments show that this indicator captures efficiently how much learning remains to be done.
– `BelVar`: the variance on the belief state: `BelVar`$= \sum_{k=1}^{n} w_k \sum_{i\in S}(b_k(i) - \hat{b}(i))^2$, where $\forall i, \hat{b}(i) = \sum_{k=1}^{n} w_k b_k(i)$. This heuristic has a good performance but has more noise than Variance.
– `InfoGain`: the quantity of information that a query could bring. Its definition is [3]: `infoGain`$= \sum_{k=1}^{n}[w_k \sum_{i,j\in S^2}[B_t(i,j)(\frac{1}{\sum_{k'\in S}\alpha^t_{A,i,k'}} + \frac{1}{\sum_{k'\in Z}\alpha^z_{A,j,k'}})]$
It has nearly the same behavior as the *distance* heuristic. Its advantage is that it is equal to zero whenever a query wouldn't bring any information (in places where the model is already well known or certain). Since it would be a complete waste of resources to do a query in these situations, it is very useful.
– `AltStateEntropy`: this heuristic measures the entropy of the mean alternate belief. `AltStateEntropy`$= \sum_{s\in S} -[\sum_{i=1}^{N}\beta_i(s)]\log(\sum_{i=1}^{N}\beta_i(s))$
It measures how much knowledge has been lost since the last query, and how inefficient the non query update would be. Using this heuristic makes the agent able to take as much information as it can from each single query.

The most useful heuristics are alternate state entropy, information gain, and the variance on the value function. So the logical function we use in our experimental section is a combination of these three heuristics.

$$doQ = (\text{AltStateEntropy} > \varepsilon_1)\text{AND}(\text{InfoGain} > \varepsilon_2)\text{AND}(\text{Variance} > \varepsilon_3) \quad (1)$$

The first condition ensures that no query should be done if enough information about the state is possessed because of previous queries. The second condition ensures that a query will not be made if it does not bring direct information about the model, which is the case if we are in a subpart of the model we know very well. The third condition ensures that we will stop doing queries when our model uncertainty does not have any influence on the expected return.

To do the non-query update of the transition alpha-parameters, we use the *alternate transition belief* $B_t(s,s')$ which is computed according to Equation 2: it is the distribution over the transitions that could have occurred in the last time step. The *non-query learning* then updates the corresponding state transition alpha-parameters according to

[3] B_t is defined in Equation 2. Furtheremore, we consider that $\frac{1}{\sum_{k'\in S}\alpha^t_{A,i,k'}} = 0$ ($\frac{1}{\sum_{k'\in Z}\alpha^z_{A,j,k'}} = 0$) when the parameters corresponding to transitions (observations) in state i, action A, are *certain*.

Equation 3 [4]. On the other hand the observation alpha-parameters are updated proportionally to the new alternate mean belief state $\tilde{\beta}'$ defined by Equation 4 according to Equation 5.

$$\begin{aligned}
\forall s,s'\, B_t(s,s') &= \sum_{i=1}^{n} w_i \frac{[O_i]^{z}_{s',A}\,[P_i]^{s'}_{s,A}\,\beta_i(s)}{\sum_{\sigma\in S}[O_i]^{z}_{\sigma,A}\,[P_i]^{\sigma}_{s,A}} && (2)\\
\forall i \in [1\dots n]\,\forall(s,s')\,\alpha_t(s,A,s') &\leftarrow \alpha_t(s,A,s') + \lambda B_t(s,s') && (3)\\
\forall s\ \tilde{\beta}'(s) &= \sum_{i=1}^{n} w_i \beta'_i(s) && (4)\\
\forall i \in [1\dots n]\,\forall s' \in S\ \alpha_z(s',A,Z) &\leftarrow \alpha_z(s',A,Z) + \lambda\,\beta'_i(s')w_i && (5)
\end{aligned}$$

3.4 Handling non-stationarity

Since we want our algorithm to be used with non-stationary POMDPs which have parameters that change over time, our model has to weight recent experience more than older experience. To do that, each time we update one of the hyper-parameter, we multiply all the hyper-parameters corresponding to the associated multinomial distribution by $\nu \in]0;1[$, which is the *model discount factor*.

This does not change the most likely value of any of the updated parameters:

$$\forall i = 1,\dots N, \theta^*_{i,new} = \frac{\nu\alpha_i}{\sum_{k=1}^{N}\nu\alpha_k} = \frac{\alpha_i}{\sum_{k=1}^{N}\alpha_k} = \theta^*_{i,old}$$

However it diminishes the confidence we have in them. Note that the *equilibrium confidence* has the following expression: $C_{max} = \lambda\frac{1}{1-\nu}$. This confidence is reached after an infinite number of samples and is an indicator of the trust we have in overall past experience. Therefore it should be high when we believe the model is stable.

Table 1 provides a detailed description of MEDUSA+, including the implementation details.

4 Theoretical properties

In this section we study the theoretical properties of MEDUSA+. We will first present some general definitions and properties and then use them to analyze the limit of the policy executed and the convergence of MEDUSA+.

4.1 Definitions

We consider a POMDP problem with $|S|$ states, $|A|$ actions and $|Z|$ observations. We call $\mathcal{B} = \{[0;1]^{|S|}\}$ the *belief space*. We call $\mathcal{H}$ the history space (which contains all the possible sequences of actions and observations). Let $\mathcal{D}_t$ be the set of Dirichlet distributions at time step t. This set corresponds to N_α alpha-parameters: $\{\alpha_1 \dots \alpha_{N_\alpha}\}$ parameters.

[4] There is an alternative to this procedure. We can also sample a query result from the alternate transition belief distribution and thus update only one parameter. However, experimental results shows that this alternative is as efficient as the belief-weighted method.

1. Let $\lambda \in (0,1)$ be the learning rate. Initialize the necessary Dirichlet distributions. Note that we can assume that some parameters are perfectly known. For any unsure transition probability, $T^a_{s,\cdot}$, define $Dir \sim \{\alpha_1, \dots \alpha_{|S|}\}$. For any unsure observation parameter $O^a_{s,\cdot}$, define $Dir \sim \{\alpha_1, \dots \alpha_{|Z|}\}$.
2. Sample n POMDPs $P_1, \dots P_n$ from these distributions. (We typically use $n = 20$). The perfectly known parameters are set to their known values.
3. Compute the probability of each model: $\{p_{01}, \dots p_{0n}\}$.
4. Solve each model $P_i \rightarrow \pi_i, i = 1, \dots n$. We use a finite point-based approximation[9].
5. Initialize the history $h = \{\}$.
6. Initialize the belief for each model $b_1 = \dots = b_n = b_0$ (We assume a known initial belief b_0). We also initialize the *alternate* belief $\beta_1 = \dots = \beta_n = b_0$.
7. Repeat:
   (a) Compute the optimal actions for each model: $a_1 = \pi_1(b_1), \dots a_n = \pi_n(b_n)$.
   (b) Randomly pick and apply an action to execute, according to the model weights: $a_i = \pi_i(b_i)$ is chosen with probability w_i where $\forall i \; w_i = \frac{p_i}{p_i}$. p_i is the *current* probability that model i has according to the Dirichlet distribution. Note that there is a modification compared to the original MEDUSA algorithm here. The reason for this particular modification is to make Proposition 2 verified.
   (c) Receive an observation z.
   (d) Update the history $h = \{h, a, z\}$
   (e) Update the belief state for each model: $b'_i = b_i^{a,z}, i = 1..n$. We also update the alternate belief β according to the action/observation pair.
   (f) Determine if we need a query or not. See Equation 1.
   (g) If the query is made, query the current state, which reveals s and s'. Update the Dirichlet parameters according to the query outcome:
   $\alpha(s, a, s') \leftarrow \alpha(s, a, s') + \lambda$
   $\alpha(s', a, z) \leftarrow \alpha(s', a, z) + \lambda$
   Multiply each of the $\alpha(s, a, *)$ parameters by the *model discount factor* ν. Set the alternate belief β so that the confidence of being in state s' is now 1.
   (h) If the query is not made, we use a non query update according to equations 3 and 5. Note that the alternate beliefs are used in this procedure.
   (i) Recompute the POMDP weights: $\{w'_1, \dots w'_n\}$.
   (j) At regular intervals, remove the model P_i with the lowest weight and redraw another model P'_i according to the current Dirichlet distribution. Solve the new model: $P'_i \rightarrow \pi'_i$ and update its belief $b'_i = b_0^h$, where b_0^h is the belief obtained when starting in b_0 and seeing history h. The same is done with the alternate belief β: we compute the belief obtained when we start from the last query and see the history.
   (k) At regular intervals, reset the problem, sample a state from the initial belief and set every b and β to b_0.

Table 1. The MEDUSA+ algorithm

Proposition 1. *Let $\mathcal{P}$ be the ensemble containing all the possible POMDP models m with $|S|$ states, $|A|$ actions and $|Z|$ observations. For any subset P of $\mathcal{P}$, we can estimate the probability $p(m \in P|\mathcal{D}_t)$. It is:*

$$p(m \in P|\mathcal{D}_t) = \int_{p \in P} \frac{\prod_{i=1}^{|S|} \theta_{i,p}^{\alpha_{<i,\mathcal{D}_t>}-1}}{F(\mathcal{D}_t)} dp, \text{ where } F(\mathcal{D}_t) = \frac{\prod_{i=1}^{|S|} \Gamma(\alpha_{<i,\mathcal{D}_t>})}{\Gamma(\sum_{i=1}^{|S|} \alpha_{<i,\mathcal{D}_t>})}.$$

This actually defines the ***measure*** $\mu : P \subset \mathcal{P} \to [0;1]$.

Proof. (1) $\forall P \subset \mathcal{P}\, \mu(P) \geq 0$. (2) $\mu(\emptyset) = 0$. (3) Let $P_1, P_2, \dots, P_n$ be disjoint subsets of $\mathcal{P}$: we have $\mu(\bigcup_i P_i) = \sum_i \mu(P_i)$. This a straightforward application of the Chasles relation for converging integrals.

Definition 1. *We say that a POMDP is* ***fully-explorable*** *if and only if:*

$$\forall s' \in S \forall a \in A\, \exists s \in S \text{ such that } p(s_t = s' | s_{t-1} = s, a_{t-1} = a) \geq 0$$

This property is verified in most POMDPs of the literature[5]. Note that we have to introduce the fact that a given state can be reached at least once by every possible action only because according to the general definition, the observation probabilities are a function of the resulting state *and* of the chosen action.

Definition 2. *Let $\Pi : \mathcal{H} \times \mathcal{P} \to A$ be the* ***policy*** *function which gives the optimal action for a given history and a given model. Note that for a given history the policy function is constant per intervals over the space of models.*

Definition 3. *A POMDP reward structure is* without coercion *if the reward structure is such that: $\forall a \in A \forall h \in \mathcal{H}\, \exists m \in \mathcal{P} s.t. \Pi(h,m) = a$*

This property is verified in most POMDP problems of the literature.

4.2 Analysis of the limit of the policy

In this subsection we analyze how the policy converges when we have an infinite number of models.

Proposition 2. *Given the history h, if we have N_m models $m_1 \dots m_{N_m}$ and every model has a weight $w_i \geq 0$ and the weights are such that $\sum_i^{N_m} w_i = 1$, we consider the following stochastic policy Ψ in which the probability of doing action a is equal to: $\sum_{i=1}^{N_m} \delta(a, \Pi(h, m_i)) w_i$ (where δ is such that $\delta(n,n) = 1$, $n \neq m \Rightarrow \delta(n,m) = 0$, and Π is the policy function, defined according to Definition 2).*

Suppose we have an infinite number of models, which are redrawn at every step. Then the policy Ψ is identical to the policy Ψ_∞ for which the probability of doing action a is:

$p(\Psi_\infty, A = a) = \int_\mu^{\mathcal{P}} \delta(a, \Pi(h,m)) dm$

$\int_\mu$ being the integral defined by the measure μ that was described in Proposition 1.

Proof sketch. Actually if the conditions are verified, the functions that define Ψ are rectangular approximations of the integral. According to the Riemann definition of the integral (as a limit of rectangular approximations or Riemann sums), these approximations always converge to the value of the integral.

[5] The property is true in the *tiger* POMDP. However it is a very strong condition and we might want to consider only *subparts* of a POMDP where this condition is true.

4.3 Convergence of MEDUSA+

In this subsection we do not consider initial priors or any previous knowledge about the parameters. We study the case in which the model is stationary (and $\nu = 1$). The intuitive idea for this proof is quite straightforward. We will show that convergence is obtained provided every state-action pair is queried infinitely often. Then we will show how our algorithm allows this condition to be true.

Theorem 1. *If a query is performed at every step, the estimation for the parameters converges with probability 1 to the true values of the parameters if (1) For a given state, every action has a non-zero probability of being taken. (2) The POMDP is **fully explorable**.*

Proof. The estimation we have for each parameter is equal to the maximum likelihood estimation (MLE) corresponding to the samples seen since the beginning of the algorithm. ($\forall i \in [1 \ldots N]\, \theta_i^* = \frac{\alpha_i}{\sum_{k=1}^{N} \alpha_k}$). We know that maximum likelihood estimation always converges to the real value when there is an infinite number of samples. Furthermore, conditions 1 and 2 assure that there will be an infinite number of them.

Theorem 2. *If the POMDP is **fully explorable**, if queries are performed at every step, if the reward structure is **without coercion** and if Ψ_∞ is followed, the parameters will converge with probability 1 to their true values given an infinite run.*

Proof. We have to verify the conditions of theorem 1. Condition 2 is verified since we have the fully-explorable hypothesis. We will prove condition 1. Let us suppose that the system is in state s after having experienced history h. The no-coercion assumption implies that for any action a there exists a model m such that $\Pi(h,m) = a$. Since the policy function is constant per intervals, there exists an interval P, subpart of $\mathcal{P}$ such that: $\forall m \in P\, \Pi(h,m) = a$ and $\int_\mu^P dm > 0$.

$p(\Psi_\infty, A = a) = \int_\mu^{\mathcal{P}} \delta(\Pi(h,m),a) dm$
$p(\Psi_\infty, A = a) > \int_\mu^{P} \delta(\Pi(h,m),a) dm$
$p(\Psi_\infty, A = a) > \int_\mu^{P} dm > 0$

So the probability of doing action a is strictly positive, which proves condition 1 and therefore proves the convergence.

According to property 2 and to theorem 2, the MEDUSA+ algorithm converges with probability 1 given an infinite number of queries, an infinite number of models, plus queries and redraws at every step. Note that we could also modify MEDUSA+ so that every action is taken infinitely often[6]. In that case the no coercion property would not be required.

Theorem 3. *The non-query version of the MEDUSA+ algorithm converges under the same conditions as above, provided that we use* `InfoGain`> 0, `AltStateEntropy` > 0, *or an AND of these.*

[6] This would imply the use of an *exploration rate*: we would have a probability of ε of doing a random action, which is a common idea in Reinforcement Learning.

Proof. If `InfoGain`$=0$ doing the query has no effect on the learned model. Furthermore if `AltStateEntropy`$=0$ it means that we can determine with full certainty what the result of the query would be if we did one: so doing the non-query update has exactly the same result as doing the query update in both cases. Therefore, using the non-query learning in MEDUSA+ with these heuristics is equivalent to performing a query at every step. So Theorem 2 can be applied, which proves the convergence.

5 Experimental results

In this section we present experimental results on the behavior of MEDUSA+. Experiments are made on the *tiger* problem described in [7]. In this framework there are two doors and a tiger behind one of them: we have two states stating where the tiger is (tiger-left and tiger-right) and three actions (hear, open-left, and open-right). The observations are hear-left and hear-right. Transitions probabilities are set such that the tiger is with equal chances behind each door and does not move when the hear action is performed. The hear action brings the correct information with probability p and the rewards are set such that opening the door with the tiger is heavily penalized, and opening the other one has a high value.

We study the behavior of MEDUSA+ in cases where we have learned a model with high confidence and we experience a sudden change in the parameter p: in this way we see how MEDUSA+ can adapt to a change in the environment.

On Figure 1, we see the behavior of the algorithm when the equilibrium confidence is low (equal to 100): the agent quickly adapts to the changed parameter value even if the variation is big. However the low confidence introduces some oscillations in the parameter values and therefore in the policy. In figure 2, we see what happens when the

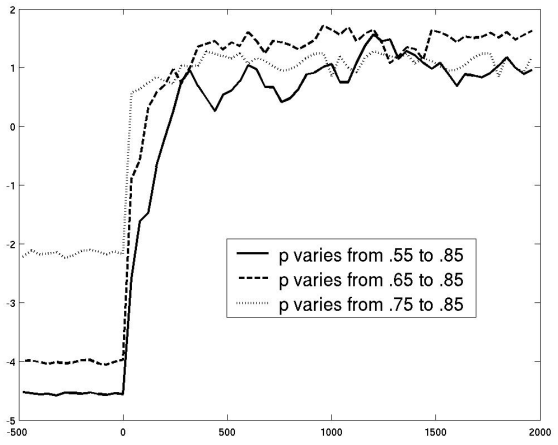

Fig. 1. Evolution of the discounted reward. There is a sudden change in the parameter p at time 0. Here, the equilibrium confidence is low (100).

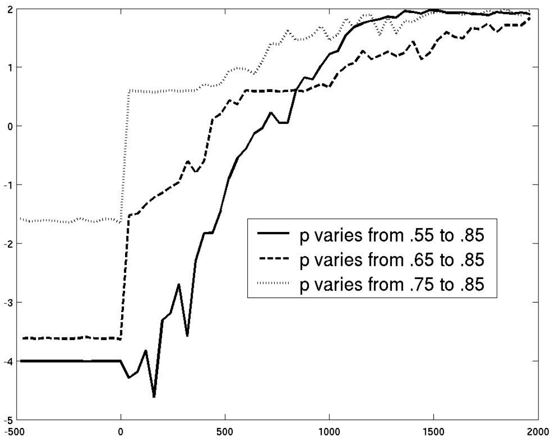

Fig. 2. Evolution of the discounted reward. There is a sudden change in the parameter p at time 0. Here, the equilibrium confidence is high (1000).

equilibrium confidence is high (equal to 1000): the agent takes more time to adapt to the new parameter value but the oscillations are less important.

These results show that we can actually tune the confidence factors to make them adapted either to almost constant parameters or to quickly changing parameters. Note that on most of these experiments we need a number of queries which is between 300 and 500. The non-query learning introduced in this paper reduces the number of queries by a factor of 3. The queries would not be needed if the changes in the parameters were small enough.

6 Discussion

Chrisman [3] was among the first to propose a method for acquiring a POMDP model from data. Shatkay & Kaelbling [11] used a version of the Baum-Welch algorithm to learn POMDP models for robot navigation. Bayesian exploration was proposed by Dearden et al. [5] to learn the parameters of an MDP. Their idea was to reason about model uncertainty using Dirichlet distributions over uncertain model parameters. The algorithm we present in Section 3 can be viewed as an extension of this work to the POMDP framework, though it is different in many respects, including how we trade-off exploration vs exploitation, and how we take non-stationarity into account.

We also point out that our work bears resemblance to some of the recent approaches for handling model-free POMDPs [8];[2];[12]. Whereas these approaches are well suited to domains where there is no good state representation, our work does make strong assumptions about the existence of an underlying state. This assumption allows us to partly specify a model whenever possible, thereby making the learning problem much more tractable (e.g., at most 1000 times fewer examples are needed.).

The other key assumption we make, which is not used in model-free approaches, regards the existence of an oracle (or human) for correctly identifying the state following each query. This is a strong assumption; however, we precise that the query result needs not be known immediately so this hypothesis is realistic: (in the tiger problem, for example, it is logical to suppose that we can tell the agent if the door it has opened had indeed the tiger behind it after the opening action is done).

Our active learning strategy handles cases where we don't know the full domain dynamics and cases where the model changes over time. Those were cases that no other work in the POMDP literature had handled.

We have explained the MEDUSA+ algorithm and the improvements we made from MEDUSA. We have discussed the theoretical properties of MEDUSA+ and presented the conditions needed for its convergence. Furthermore, we have shown empirical results showing that MEDUSA+ can cope with non stationary POMDPs with a reasonable amount of experimentation.

References

[1] Anderson, B. and Moore, A. "Active Learning in HMMs" 2005.

[2] Brafman, R. I. and Shani, G. "Resolving perceptual aliasing with noisy sensors" NIPS, 2005.

[3] Chrisman, L. "Reinforcement learning with perceptual aliasing: The perceptual distinctions approach" Proceedings of the Tenth International Conference on Artificial Intelligence, pages 183–188, AAAI Press, 1992.

[4] Cohn, D. A., Ghahramani, Z. and Jordan, M. I. "Active Learning with Statistical Models" NIPPS, 1996.

[5] Dearden, R.,Friedman, N.,Andre, N., "Model Based Bayesian Exploration" Proc. Fifteenth Conf. on Uncertainty in Artificial Intelligence, 1999.

[6] Jaulmes, R.,Pineau, J.,Precup, D., "Active Learning in Partially Observable Markov Decision Processes" ECML, 2005.

[7] Kaelbling, L., Littman, M. and Cassandra, A. "Planning and Acting in Partially Observable Stochastic Domains" Artificial Intelligence. vol.101, 1998.

[8] McCallum, A. K. "Reinforcement Learning with Selective Perception and Hidden State" Ph.D. Thesis. University of Rochester, 1996.

[9] Pineau, J., Gordon, G. and Thrun, S. "Point-based value iteration: An anytime algorithm for POMDPs" IJCAI, 2003.

[10] Poupart, P. and Boutilier, C. "VDCBPI: an Approximate Scalable Algorithm for Large Scale POMDPs" NIPS, 2005.

[11] Shatkay, H., Kaelbling, L. "Learning topological maps with weak local odometric information" Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (pp. 920–927). Morgan Kaufmann, 1997.

[12] Singh, S., Littman, M., Jong, N. K., Pardoe, D., and Stone, P. "Learning Predictive State Representations" ICML, 2003.

[13] Spaan, M. T. J. Spaan, and Vlassis, N. "Perseus: randomized point-based value iteration for POMDPs". Journal of Artificial Intelligence Research, 2005. To appear.