V. Kůrková et al. (Eds.): ITAT 2014 with selected papers from Znalosti 2014, CEUR Workshop Proceedings Vol. 1214, pp. 21–27
http://ceur-ws.org/Vol-1214, Series ISSN 1613-0073, © 2014 R. Brunetto, M. Vomlelová

Acting and Bayesian reinforcement structure learning of partially observable environment

Robert Brunetto, Marta Vomlelová

Charles University, Czech Republic
robert@brunetto.cz

Abstract: This article shows how to learn both the structure and the parameters of partially observable environment simultaneously while also online performing near-optimal sequence of actions taking into account exploration-exploitation tradeoff. It combines two results of recent research: The former extends model-based Bayesian reinforcement learning of fully observable environment to bigger domains by learning the structure. The latter shows how a known structure can be exploited to model-based Bayesian reinforcement learning of partially observable domains. This article shows that merging both approaches is possible without too excessive increase in computational complexity.

1 Introduction

Partially observable Markov decision processes (POMDPs) are a well known framework for modeling and planning in partially observable stochastic environments [13], [8].

Such plans could be used in robotics, in dialogue management or elsewhere. However, the range of their practical applicability is limited by several difficulties. Firstly, their usage is limited to small state spaces only (curse of dimensionality) even when using several approximations [9].

Secondly, it could be difficult for an expert to specify the probabilities exactly when applying POMDP to some domain. There were several attempts to overcome this problem by learning the model [7], [3], [4] or applying other techniques [2], [6].

Following the recent research this article proposes an approach which could solve both problems at the same time.

Learning the model by interacting with the environment (reinforcement learning) was researched a lot [1]. During the last decade Jaulmes and Pineau tried to learn unfactorized POMDP [7] by assuming a prior probability distribution over all possible models. Then Poupart and Vlassis [11] used similar Bayesian reinforcement learning technique to learn the parameters of a DBN [1] with the known structure.

If the structure were not known then learning it would be a challenging problem even in fully observable environments. Ross and Pineau [10] addressed it and proposed an online algorithm for it.

This article addresses generalization of the tasks mentioned above. We show the way to navigate an agent in an unknown stochastic dynamic partially observable environment using near optimal actions. To do so, we combine the algorithm [11] for learning parameters of POMDP with the known structure with the algorithm [10] for learning the structure of MDP.

We take the best ideas of each of them. Right after explaining the basic preliminaries in section 2 and demonstrating them on an example in section 3 we present an analytical representation of the belief over parameters (analogous to [11]) in section 4. Similarly to the article [10] the belief over structures will be maintained by MCMC[2]

algorithm described in section 5. This allows us to use an online action selection procedure described in section 7. In the last section we review the approximations from [11] which can be also applied in our settings making our model more traceable and compact.

2 Preliminaries

POMDPs usually model unknown environments as follows. At each time step the environment with the agent is in an unobserved state $\mathbf{s} \in S$. At this time step agent receives observation $\mathbf{o} \in O$ and reward $\mathbf{r}$. Then the agent is free to choose an action $a \in A$. As the time advances, the state changes to a new state $\mathbf{s}'$ that stochastically depends on previous state and action.

We are interested in the special case of POMDPs where the state space factorizes along several state variables $X \in \mathbf{X}$ hence each state $\mathbf{s}$ is a vector $\mathbf{s} = (s_1, \ldots s_{|\mathbf{X}|}) \in S = \prod_{X \in \mathbf{X}}$.

Rewards and observations behave analogously $\mathbf{r} \in \prod_{R \in \mathbf{R}}$ and $\mathbf{o} \in \prod_{O \in \mathbf{O}}$.

Moreover we can assume without loss of generality that $\mathbf{O} \subseteq \mathbf{X}$ and $\mathbf{R} \subseteq \mathbf{X}$. It implies that the observation is the state restricted to observation variables. It will be denoted as $\mathbf{o} = \mathbf{s}_{\mathbf{O}}$. Other restriction operations will be denoted analogously. For an instance, values of reward variables in the state $\mathbf{s}$ can be denoted $\mathbf{s}_{\mathbf{R}}$.

We will always restrict vectors which are denoted by bold lower case letters to sets of variables denoted as bold upper case letters. Capital letter which won't be bold will denote single variable. Lower case non-bold letter will be its value. Specially in the case when G is

[1]DBN stands for Dynamic Bayesian Network.

[2]MCMC stands for Markov Chain Monte Carlo.

the graph of a Bayesian network, $\mathbf{PA}_G X$ we will denote set of variables which are parents[3] of variable X in graph G. Values of variables of parents of X will be denoted $\mathbf{pa}_G X$ where $X \in \mathbf{X}$.

When the time advances to the next step the state $\mathbf{s}$ changes to $\mathbf{s}'$ according to unknown transition probability $P(\mathbf{s}'|\mathbf{s},a)$.

We assume that the transition probability $P(\mathbf{s}'|\mathbf{s},a)$ factorizes according to a dynamic Bayesian network with unknown structure G with unknown parameters.

$$P(\mathbf{s}'|\mathbf{s},a) = \prod_{X\in\mathbf{X}} P(\mathbf{s}'_X|(\mathbf{s}',\mathbf{s},a)_{\mathbf{PA}_G X})$$

Dynamic Bayesian network with structure G is described as Bayesian network which has $\mathbf{X}' \cup \mathbf{X} \cup \{\mathbf{A}\}$ as its nodes. By $(\mathbf{s}',\mathbf{s},a)_{\mathbf{PA}_G X}$ we denote $(\mathbf{s}',\mathbf{s},a)$ restricted to the variables which are parents of variable X according to structure G.

To emphasize that we take $P(\mathbf{s}'_X|(\mathbf{s}',\mathbf{s},a)_{\mathbf{PA}_G X})$ as an unknown parameter, we will denote it:

$$\theta_X^{(\mathbf{s}',\mathbf{s},a)_{\mathbf{PA}_G X}}. \tag{1}$$

It is actually a vector of real values between 0 and 1 summing to 1 containing for each value $\mathbf{s}'_X$ the probability $P(\mathbf{s}'_X|(\mathbf{s}',\mathbf{s},a)_{\mathbf{PA}_G X})$. As the indexes suggest that we have such a set of parameters for each state variable X and for each combination of values of its parents.

Even though we do not assume structure G to be known we still assume that a prior probability distribution $P(G)$ over structures G is known. This probability distribution can express expert's prior knowledge or it can just prefer simple structures over more complicated ones.

The transition probability can be expressed as:

$$P(\mathbf{s}'|\mathbf{s},a) = \sum_G P(G)\cdot P(\mathbf{s}'|\mathbf{s},a,G). \tag{2}$$

We assume that the parameters $\theta_X^{(\mathbf{s}',\mathbf{s},a)_{\mathbf{PA}_G X}}$ follow the Dirichlet distribution, which is conjugate to multinomial distribution.

The hyperparameters are initially set to one or set to the values that preserve the likelihood of equivalent parameters in equivalent Bayesian networks (see [5]).

Each unknown parameter $\theta_X^{(\mathbf{s}',\mathbf{s},a)_{\mathbf{PA}_G X}}$ can be regarded as an additional state feature.

This way a POMDP with unknown parameters can be converted into a bigger POMDP without unknown parameters. Similar remark also holds for the unknown structure. It can be also regarded as a part of the state hence POMDP with unknown structure and parameters can be regarded as a bigger POMDP with known dynamics.

POMDP agent is usually maintaining so called belief state $b(\tilde{\mathbf{s}})$ which represents probability distribution over possible states. In our case $\tilde{\mathbf{s}}$ consists not only of the current state $\mathbf{s}$ but also of the graph structure and its parameters.

Surprisingly, as Poupart and Vlassis showed [11], even though there is an infinite number of possible parameters the belief for a given structure can be maintained in a closed form. We will review it in section 4.

There is only a finite number of possible graphical structures implying that the whole belief $b(\tilde{\mathbf{s}})$ can be maintained in a closed form. But the number of possible graphs may be large. That is why approximation is introduced in section 5.

The belief is a sufficient statistic for taking a decision. Agent's overall algorithm is summarized in algorithm 1.

Algorithm 1: Agent's life cycle

```
Data: $b$ initial belief over current state, structures and
      their parameters
1 while not TerminationCondition() do
2     $a$ := selectAction($b$).action;
3     performAction($a$);
4     $\mathbf{o}$ := receiveObservation();
5     $b$ := updateBelief($b$,$a$,$\mathbf{o}$);
```

3 Example

For better readability we illustrate the terms on a problem taken from [8]. Imagine the following scenario. The agent is before two doors. Behind one of the doors is a fortune which would give the agent big reward but behind the second door is a tiger which could cause the agent even higher negative reward. The agent doesn't know which door is which. It has three possibilities what to do. It can walk through one door, walk through second door or wait and listen behind which door is bigger noise and then face the same decision with the newly gained information.

This corresponds to algorithm 1. Choosing and performing action is on its lines 2 and 3. After the agent listens or opens a door it knows what it has heard or whether it met a tiger or found a fortune. This is shown on line 4 of algorithm 1. Agent then uses this observation to update its belief on line 5. The belief contains all information the agent knows i.e. the agent should count how many times it has heard the the tiger on each side and estimate where the tiger is. It should also learn usual tiger position and other facts about the environments' behaviour. These information are also stored and updated with belief. We assume that after opening a door and finding a tiger or a fortune the experiment restarts itself and the tiger is again

[3]Parent variable V of variable W is a term used in Bayesian-networks-literature to denote that there is an arrow from V to W in the graph of Bayesian network. This is in greater detail explained in section 3.

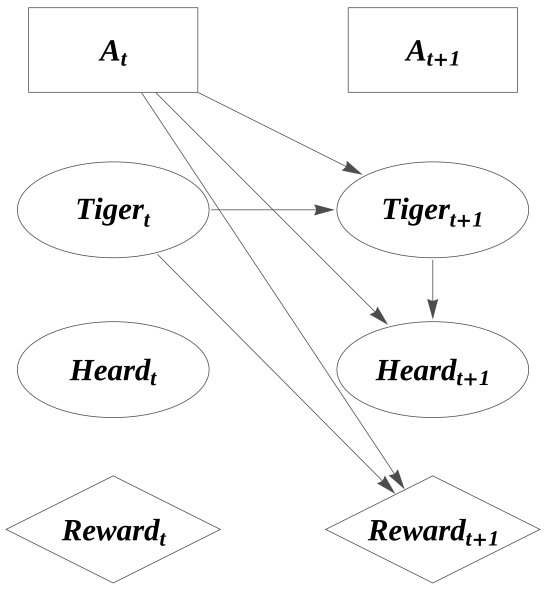

Figure 1: The tiger example

placed behind random door. Everything repeats until a termination condition is reached (line 1). This unspecified condition allows for example infinite repetition of the experiment until the user terminates it.

This scenario can be modelled as follows. The set of actions A contains actions for going left, right and listening.

$$A \in \{left, right, listen\}$$

The set of reward variables $\mathbf{R}$ contains only one variable, namely the variable $Reward$.

$$\mathbf{R} = \{Reward\}$$

We assume that domains of all variables are known. Hence the domain of variable $Reward$ is also assumed to be known. Let us say that cost of listening is 1, cost of facing a tiger 100 and reward for finding a fortune 10. Then $Reward \in \{-1, -100, 10\}$.

The set of state variables $\mathbf{X}$ contains variables $Tiger$, $Heard$ and $Reward$.

$$\mathbf{X} = \{Tiger, Heard, Reward\}$$

$Tiger$ is position of the tiger and $Heard$ is a variable containing what has the agent heard.

From this three variables the variable Tiger is unobserved because agent doens't know the position of the tiger. Other two variables are observed.

$$O = \{Heard, Reward\}$$

The agent knows neither the state transition probabilities nor the structure between the variables. It has to learn it first. The real graphical structure could be for example like figure 1.

The node shapes have its usual meaning. Squares denote the decision(action) variables. Circles denote random variables and diamonds denote reward variable.

As we are modelling a dynamic environment. We have a copy of each variable for each time step. The figure shows only two time slices - for time t and $t+1$. We have also shown only the arrows which point to time t+1. We have omitted all other arrows.

An arrow starts in a node called parent and goes to a node called child. As each node is a variable we interchange the term parent node and parent variable.

The intuitive meaning of arrows is this: The child node(variable) X stochastically depends only on combination of values of parent variables. More rigorous meaning of an arrow is that child variable X is conditionally independent on all other variables given values $\mathbf{pa}_G X$ of its parents $\mathbf{PA}_G X$.

Let G denote the graph from figure 1. Then $\mathbf{PA}_G Reward_{t+1} = \{A_t, Tiger_t\}$ which is quite natural. The reward which agent will receive depends on the combination where it decided to go and where the tiger has been. This dependence in our example is even deterministic. However this is not general.

For example the variable $Heard_{t+1}$ depends on its parents stochastically. The reason for it is simple. The agent could independently of the current state randomly hear incorrectly. $\mathbf{PA}_G Heard_{t+1} = \{A_t, Tiger_t\}$ because what the agent hears depends on where the tiger is and on the fact whether the agent even listed.

In order to make it possible for the agent to learn the environment it is necessary to let the agent interact with the environment repeatedly. So the experiment with tiger restarts itself everytime the agent opens a door. The tiger is then again placed behind a random door. That is why $\mathbf{PA}_G Tiger_{t+1} = \{A_t, Tiger_t\}$.

Each variable in this example has two parents: the ternary action variable and one binary variable. Hence to specify the probability distribution over child variable we need to specify the probability of each of its values for all six combinations of values of parent variables ie. we need to specify $P(\mathbf{s}'_X | (\mathbf{s}', \mathbf{s}, a)_{\mathbf{PA}_G X})$. It is specified by $2 * 3 = 6$ probability distributions over values of child variable. As all child variables are binary we need to specify 6 pairs of numbers between 0 and 1 such that each pair sums to 1.

These pairs of numbers are denoted as in (1).

We interchangeably call such pairs or even above mentioned 6-tuples of pairs parameters in plural or a parameter in singular. By substituting values to $\mathbf{s}$, $\mathbf{s}'$ and a one can restrict the parameter to one concrete value.

For example let us assume that the agent is listening. The tiger is behind the right door and that the probability of correct listening is 0.9 and the probability of opposite listening outcome is 0.1 then following equalities would hold:

$$P(\mathbf{s}'_{Heard} | (\mathbf{s}', \mathbf{s}, a)_{\mathbf{PA}_G Heard}) = \theta_{Heard}^{(right, listen)} = (0.9, 0.1).$$

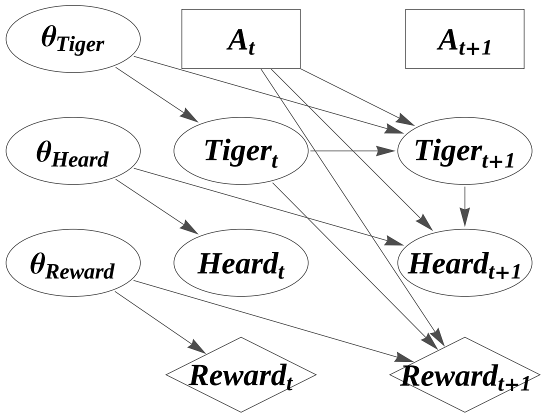

Figure 2: The tiger example

The first equality holds because

$$\mathbf{pa}_G Heard = (\mathbf{s}', \mathbf{s}, a)_{\mathbf{PA}_G Heard} = (right, listen).$$

The agent doesn't know the value of mentioned parameter and as was already explained in previous section the agent can treat parameters as random variables. This converts the POMDP with unknown parameters to a bigger POMDP with known parameters. The structure of this POMDP is shown in figure 2.

Notice that we don't have a copy of these parameter-random-variables for each time slice. All time slices share the same parameter. That is because the parameter doesn't change over time.

Information about possible values of these parameters, along with possible structures and states are maintained in the belief, which is described in following sections.

4 Belief for a given structure

We begin by describing how belief looks like when we are given the structure. It is exactly the same as described by Poupart & Vlassis [11].

The key component is the nice properties of Dirichlet distribution. Let us have one discrete random variable V which can take values from 1 up to K with probabilities $(\theta_1, \theta_2, \ldots\theta_K)$, where $\sum_i \theta_i = 1$.

The usual way to estimate these parameters in Bayesian statistics is to compute the posterior when assuming that the prior follows dirichlet distribution. Its density is given by $D(\theta;\mathbf{n}) = \frac{1}{B(\mathbf{n})}\prod_i \theta_i^{n_i-1}$, where $\theta = \{\theta_i\}_i$, $\mathbf{n} = \{n_i\}_i$ are some hyperparameters and $B(\mathbf{n})$ is a constant depending on them which makes the distribution sum to one. It is known as multinomial beta function.

The prior distribution which states that we have no evidence is expressed by setting all hyperparameters n_i equal to 1. It can be interpreted as evidence that all values were observed exactly once or it can be thought of only as smoothing the posterior.

The posterior probability that the variable V contains value i is then equal to proportion of how many times was value i observed.

$$P(V = i) \quad = \quad \frac{n_i}{\sum_i n_i}. \qquad (3)$$

In the fully observable environment then we could estimate all parameters in the whole structure by (3). We would use this estimate for each state variable X and for each combination of values of its parents $\mathbf{pa}_G X$. That is the reason why we add X as a lower index to θ and $\mathbf{pa}_G X$ as an upper index to θ as in (1).

Each set of parameters $\theta_X^{\mathbf{pa}_G X}$ sums to one.

From now on θ without any indexes will denote all these sets of parameters together.

In this simplified case when the whole history was observed the density of probability of being in "information state" θ is

$$\prod_{X,\mathbf{pa}_G X} D(\theta_X^{\mathbf{pa}_G X}; \mathbf{n}_X^{\mathbf{pa}_G X})$$

The problem that not all state features are observable can be overcome by the following theorem proven by Poupart and Vlassis [11].

Theorem 1. *If the prior is a mixture of products of Dirichlets*

$$b(\mathbf{s}, \theta) = \sum_i c_{i,\mathbf{s}} \prod_{X',\mathbf{pa}_G X'} D(\theta_{X'}^{\mathbf{pa}_G X'}; \mathbf{n}_{X',i,\mathbf{s}}^{\mathbf{pa}_G X'}) \qquad (4)$$

then the posterior is also a mixture of products of Dirichlets

$$b_{a,o'}(\mathbf{s}', \theta) = \sum_j c_{j,\mathbf{s}'} \prod_{X',\mathbf{pa}_G X'} D(\theta_{X'}^{\mathbf{pa}_G X'}; \mathbf{n}_{X',j,\mathbf{s}'}^{\mathbf{pa}_G X'}).$$

Because the proof of the theorem 1 actually gives us the algorithm to update the belief we repeat the proof in this article but for better readability we split it in two lemmas.

The first lemma is technical observation which says that multiplication by $\theta_X^{\mathbf{pa}_G X}$ only scales the Dirichlet distribution.

Lemma 1. *Let $\theta = (\theta_1, \ldots\theta_K)$ and $\mathbf{n} = (n_1, \ldots, n_K)$ then*

$$\theta_j D(\theta;\mathbf{n}) = cD(\theta;\mathbf{m})$$

for some constant c and vector $\mathbf{m}$.

Proof.

$$\begin{aligned}
\theta_j D(\theta;\mathbf{n}) &= \frac{1}{B(\mathbf{n})}\theta_j \prod_i \theta_i^{n_i-1} = \\
&= \frac{1}{B(\mathbf{n})}\prod_i \theta_i^{n_i-1+[i==j]} = \\
&= \frac{B(\mathbf{n}+\delta)}{B(n)} D(\theta;\mathbf{n}+\delta) = \\
&= \frac{n_j}{\sum_{i=1}^K n_i} D(\theta;\mathbf{n}+\delta)
\end{aligned}$$

where $\delta = (0,...1,...0)$ is a vector of zeroes with one in j-th position.

The last equality follows from the property of B($\mathbf{n}$) saying that $B(\mathbf{n}) = \frac{\prod_{i=1}^{K}\Gamma(n_i)}{\Gamma(\sum_{i=1}^{K} n_i)}$. □

Lemma 2. *$P(\mathbf{s}'|\mathbf{s},a,\theta)b(\mathbf{s},\theta)$ is mixture of products of Dirichlets.*

Proof.

$$P(\mathbf{s}'|\mathbf{s},a,\theta)b(\mathbf{s},\theta) =$$

$$= (\prod_{X\in\mathbf{X}} \theta_X^{(\mathbf{s}',a,\mathbf{s})_{\mathbf{PA}\ X}})b(\mathbf{s},\theta) \tag{5}$$

$$= (\prod_{X\in\mathbf{X}} \theta_X^{(\mathbf{s}',a,\mathbf{s})_{\mathbf{PA}\ X}})\sum_i c_{i,\mathbf{s}} \prod_{X,\mathbf{pa}X} D(\theta_X^{\mathbf{pa}X}; n_{X,i,\mathbf{s}}^{\mathbf{pa}X})$$

$$= \sum_i c_{i,\mathbf{s}} \prod_{X\in\mathbf{X}} \theta_X^{(\mathbf{s}',a,\mathbf{s})_{\mathbf{PA}\ X}}) \prod_{\mathbf{pa}X} D(\theta_X^{\mathbf{pa}X}; n_{X,i,\mathbf{s}}^{\mathbf{pa}X}) \tag{6}$$

$$= \sum_i c_{i,\mathbf{s}}^{\mathbf{s}'} \prod_{X,\mathbf{pa}X} D(\theta_X^{\mathbf{pa}X}; n'^{\mathbf{pa}X}_{X,i,\mathbf{s}}) \tag{7}$$

$$\tag{8}$$

$b(\mathbf{s},\theta)$ in the formula (5) is replaced by its the definition. Then formula (6) is received by switching the terms. Moreover thanks to lemma 1 $\theta_X^{(\mathbf{s}',a,\mathbf{s})_{\mathbf{PA}\ X}}$ can be consumed by dirichlet distribution $D(\theta_X^{\mathbf{pa}X}; \mathbf{n}'^{\mathbf{pa}X}_{X,i,\mathbf{s}})$ where $\mathbf{pa}X = (\mathbf{s}',a,\mathbf{s})_{\mathbf{PA}\ X}$.

Let c and δ be the as in lemma 1. $c_{i,\mathbf{s}}$ must be multiplied by c. That is why it was replaced by $c_{i,\mathbf{s}}^{\mathbf{s}'} = c \cdot c_{i,\mathbf{s}}$ in equation (7).

$\mathbf{n}'^{\mathbf{pa}X}_{X,i,\mathbf{s}} = \mathbf{n}^{\mathbf{pa}X}_{X,i,\mathbf{s}} + \delta$ when $\mathbf{pa}X = (\mathbf{s}',a,\mathbf{s})_{\mathbf{PA}\ X}$ and $\mathbf{n}'^{\mathbf{pa}X}_{X,i,\mathbf{s}} = \mathbf{n}^{\mathbf{pa}X}_{X,i,\mathbf{s}}$ otherwise.

□

Proof of Theorem 1.

$$b_{a,\mathbf{o}'}(\mathbf{s},\theta) = k\delta(\mathbf{s}'_{\mathbf{O}'} = \mathbf{o})\sum_{\mathbf{s}} P(\mathbf{s}'|\mathbf{s},a,\theta)b(\mathbf{s},\theta)$$

$P(\mathbf{s}'|\mathbf{s},a,\theta)b(\mathbf{s},\theta)$ is mixture of products of Dirichlets by lemma . Hence $\sum_{\mathbf{s}} P(\mathbf{s}'|\mathbf{s},a,\theta)b(\mathbf{s},\theta)$ is also mixture of products of dirichlets which are closed under multiplication by constant. It follows that $b_{a,\mathbf{o}'}(\mathbf{s},\theta)$ is mixture of products of dirichlets. □

Theorem 1 implies that the belief $b(\mathbf{s},\theta)$ over state s and information state θ can be maintained in a closed form. But how can we from this representation get the probability of being in a specific state? We show in the following theorem that this can be easily done.

Theorem 2. *θ in formula (4) for the mixture of products of dirichlet can be integrated out in a closed form.*

Proof.

$$\begin{aligned}
b(\mathbf{s}) &= \int b(\mathbf{s},\theta)d\theta = \\
&= \sum_i c_{i,\mathbf{s}} \int \prod_{X',\mathbf{pa}_G X'} D(\theta_{X'}^{\mathbf{pa}_G X'}; \mathbf{n}_{X',i,\mathbf{s}}^{\mathbf{pa}_G X'})d\theta = \\
&= \sum_i c_{i,\mathbf{s}} \prod_{X',\mathbf{pa}_G X'} \int D(\theta_{X'}^{\mathbf{pa}_G X'}; \mathbf{n}_{X',i,\mathbf{s}}^{\mathbf{pa}_G X'})d\theta = \\
&= \sum_i c_{i,\mathbf{s}}
\end{aligned}$$

The first equation defines symbol $b(\mathbf{s})$. The second follows from the definition of $b(\mathbf{s},\theta)$ by switching sum and integral. The third equation switches integral and multiplication which is possible in this case because each factor depends on the different variable of multidimensional integration. Density of all probability distributions (including Dirichlet distribution) integrates to one and product of ones is one. That is why the last equation holds. □

Despite this encouraging results the number of components in the mixture grows exponentially. Luckily the belief can be approximated by the approximation proposed by Poupart & Vlassis. This will be described in section 7. Firstly, in the next section, we describe the way the belief over possible structures is maintained.

5 Belief over structures

The simplest and most naive approach to maintaining the overall belief which contains the probability of structure, its parameters and state would be straightforward. It is sufficient to keep the belief for each structure and probability of the structure. The problem is that the number of graphs on given number of vertices grows very fast with the increasing number of vertices but we want to maintain only the small number of graphs.

We propose to remember only one randomly chosen structure where the probability that the structure G is chosen would be proportional to the probability $P(G|history)$.

The probability $P(G|history)$ could be difficult to compute. But, as Ross & Pineau noted in article [10] about MDP, a Markov chain of graphs can be maintained using Metropolis-Hastings algorithm. The algorithm will ensure that the Markov chain converges to distribution of graphs which is equal to $P(G|history)$.

The Metropolis-Hastings algorithm needs to use $P(history|G)$ which can be computed as follows: It is equal to $P(\mathbf{o}|a,b,G) \cdot P(h_{t-1}|G)$, where h_{t-1} denotes history up to previous time step.

Then the idea of computing $P(\mathbf{o}'|a,b)$ is as follows: $P(\mathbf{o}'|a,b)$ is equal (9) to sum of $P(\mathbf{s}'|a,b)$ over states $\mathbf{s}'$ compatible with observation $\mathbf{o}'$ and $P(\mathbf{s}'|a,b)$ can be computed directly from hyperparameters. For the simplicity of notation we omit conditioning on G.

More precisely it is as follows:

$P(\mathbf{o}'|a,b) =$

$$= \sum_{\substack{\mathbf{s}'\in S\\ \mathbf{s}'_O=\mathbf{o}'}} P(s'|a,b) \tag{9}$$

$$= \sum_{\substack{\mathbf{s}'\in S\\ \mathbf{s}'_O=\mathbf{o}'}} \sum_{\mathbf{s}} \int_{\theta} P(s'|a,s,\theta)b(\mathbf{s},\theta) \tag{10}$$

$$= \sum_{\substack{\mathbf{s}'\in S\\ \mathbf{s}'_{\mathbf{O}}=\mathbf{o}'}} \sum_{\mathbf{s}} \sum_{i} c^{\mathbf{s}'}_{i,\mathbf{s}} \tag{11}$$

Conditioning on b can be replaced as in (10). $P(s'|a,s,\theta)b(\mathbf{s},\theta)$ in formula (10) is a mixture of products of dirichlets by lemma 4. Hence by theorem 2 it can be replaced by sum of coefficients as in 10.

The algorithm for belief update is given in algorithm 2 where $q(G'|G)$ is the probability of transition from graph G to graph G'. This distribution can be set any arbitrary way which will ensure that all graphs are reachable. $P(G)$ is prior distribution over graph structures and the probability $P(history|G')$ can be computed as described above.

Random transitions with probability $min\left(1, \frac{P(history|G')P(G')q(G'|G)}{P(history|G)P(G)q(G'|G)}\right)$ are well known under the name Metropolis-Hastlings algorithm. This algorithm ensures that the Markov chain of graphs converges to the distribution $P(G|history)$ which implies that our algorithm eventually learns either the correct structure or a structure with is equally good with respect to encountered history.

One possible way of implementing random changes and distribution $q(G'|G)$ is local change of the graph structure which can include deleting, reversing or adding edge. If the change would be local affecting only limited number of variables then necessary changes to representation of $b_{a,o}$ will be also local.

This change depends on distribution $q(G'|G)$ and isn't explicitly written in algorithm 2. We assume that this change is done on the line $G := G'$ which changes the structure.

Algorithm 2: updateBelief(b, a, o)

```
Data: belief $b$ (containing structure G and belief for
      that given structure), action $a$, observation $o$
Result: representation of belief for the next time step
G' := random modification of G;
With a probability $min\left(1, \frac{P(history|G')P(G')q(G'|G)}{P(history|G)P(G)q(G'|G)}\right)$
    G:=G';
b := $b_{a,o}$;
Simplify representation of $b$ by an approximation
from section 7.
```

Final change of belief inside one given structure $b := b_{a,\mathbf{o}}$ can be done as in theorem 1. This change includes the changes of coefficients $c_{i,\mathbf{s}}$ and hyperparameters $\mathbf{n}^{\mathbf{pa}_G X'}_{X',i,\mathbf{s}}$.

The changes are described in the proof of theorem 1 and associated lemmas.

6 Action selection

Which action should the agent choose? It should choose the action which maximizes his expected reward with respect to the probability distribution $b(\mathbf{s})$ over the states. This expected reward can be tractably estimated by a recursive approach using depth limited Monte Carlo search.

The algorithm for each action samples several new states. Each sampled state $\mathbf{s}$ is then restricted to the observation and then used to update the belief. Its value can be then again estimated by the same algorithm using recursion up to some maximum depth.

This generates sampled walks(series of states) of equal length. Their rewards are summed and in the maximum depth some simple estimate $V(b)$ of the value of belief b is added to the estimate eg. $V(b) = \sum_{\mathbf{s}} b(\mathbf{s})reward(\mathbf{s})$ where $reward(\mathbf{s}) = \sum_{R\in\mathbf{R}} \mathbf{s}_R$.

At each level of recursion is the best action chosen. For clarity this Monte Carlo search is shown in algorithm 3.

Algorithm 3: selectAction(b, d)

```
Data: belief $b$, depth $d$
Static variable: the number of samples $N$
        Result: utility estimate of belief state and
        selected action
if d = 0 then
    return $V(b)$;
maxQ := -∞;
forall the actions $a \in A$ do
    Q := 0;
    for i=1 to N do
        $\mathbf{s}'$ := sampleState($b_a$);
        $\mathbf{o}'$ := $\mathbf{s}'_{\mathbf{O}}$;
        $b'$ := updateBelief($b, a, \mathbf{o}$);
        $Q$+= (reward($\mathbf{s}'$)+
        $+\gamma\cdot$ selectAction($b', d-1$).utility)$/N$;
    if Q > maxQ then
        maxQ := Q;
        selectedAction := a;
return (maxQ,selectedAction);
```

The state can be sampled by algorithm 4.

Algorithm 4: sampleState(b)

```
Data: b($\mathbf{s}$)
Result: sampled state $\mathbf{s}$
return state sampled according to weights $\sum_i c_{i,\mathbf{s}}$
```

Firstly, it draws a graph according to graph posterior. Then it randomly draws a state with probabilities $\sum_i c_{i,\mathbf{s}}$. Which is correct as can be seen thanks to theorem 2.

7 Approximate representation of the mixture of products of Dirichlets

As noted in section 4 the number of components of the mixture representing current belief for a given graph grows exponentially with time which is untraceable. Each component of the mixture is associated with coefficients $c_{i,s}$. Naturally some of these coefficients will be smaller while the others will be bigger. We propose to handle this approximately and keep only some components with bigger coefficients.

There are several possibilities:

1. Keep k components with the greatest coefficients.
2. Sample k components with coefficients used as a probability.
3. Instead of generating a lot of components and the sampling only k of them one can directly generate only these randomly chosen components.

For a more detailed description of these and other approximations along with some of their advantages and disadvantages we encourage the reader to read [11] where the same approximations are described.

8 Conclusion and Future Work

This article has shown the design of an agent. The agent can be placed to an unknown partially observable environment and it can learn its dynamics by interaction with it. During the interaction it takes into account the exploration-exploitation trade-off and chooses a near-optimal action. The dynamics of the environment learnt by our agent is represented by a dynamic Bayesian network which structure is also learnt by our model.

The algorithm could at any time return the graph of dynamic Bayesian network it is currently maintaining as the estimate of the real structure.

This article has shown a possible way of combining known techniques of reinforcement learning of structure of MDP and parameters of POMDP in order to learn both the structure and parameters of POMDP simultaneously.

I am currently implementing and testing the proposed algorithms. One thing I am going to test is the alternative representation of belief over possible structures. We proposed to maintain one Markov chain of possible graph structures. Alternatively algorithm could maintain several such Markov chains. It could lead to the possibility of more precise decisions. Another alternative is to use a particle filter [12] where each paticle is a graph of dynamic Bayesian network. I am looking forward to reporting empiric results in the near future.

Acknowledgement: This research was supported by SVV project number 260 104 and by GAČR No. P202/10/1333.

References

[1] Barto, A. G. (1998). Reinforcement learning: An introduction. MIT press.

[2] Brunetto, R. (2013). Probabilistic modeling of dynamic systems. Informacne Technologie-Aplikacie a Teoria.

[3] Doshi-Velez, F. (2009). The infinite partially observable markov decision process. In NIPS (pp. 477-485).

[4] Doshi, F., Wingate, D., Tenenbaum, J., & Roy, N. (2011). Infinite dynamic bayesian networks. In Proceedings of the 28th International Conference on Machine Learning (ICML-11) (pp. 913-920).

[5] Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. Machine learning, 20(3), 197-243.

[6] Itoh, H., & Nakamura, K. (2007). Partially observable Markov decision processes with imprecise parameters. Artificial Intelligence, 171(8), 453-490.

[7] Jaulmes, R., Pineau, J., & Precup, D. (2005). Active learning in partially observable markov decision processes (pp. 601-608). Springer Berlin Heidelberg.

[8] Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. Artificial intelligence, 101(1), 99-134

[9] Pineau, J., Gordon, G., & Thrun, S. (2011). Anytime Point-Based Approximations for Large POMDPs. arXiv preprint arXiv:1110.0027.

[10] Poupart, P., & Vlassis, N. A. (2008). Model-based Bayesian Reinforcement Learning in Partially Observable Domains. In ISAIM.

[11] Ross, S., & Pineau, J. (2012). Model-based Bayesian reinforcement learning in large structured domains. arXiv preprint arXiv:1206.3281.

[12] Russell, S., & Norvig, P. (2009). Artificial Intelligence: A Modern Approach. Prentice Hall, 3rd edition

[13] Smallwood, R. D., & Sondik, E. J. (1973). The optimal control of partially observable Markov processes over a finite horizon. Operations Research, 21(5), 1071-1088.