Robustness of Estimation of state of the world in Partially Observable Markov Decision Process (POMDP)

May 2018

0.1 Introduction

Partially observable Markov decision processes (POMDPs) provide a natural model for sequential decision making problems where effects of actions are nondeterministic and the state of the world is not known with certainty. This model augments a well-researched framework of Markov decision processes (MDPs) [3][1] to situations where an agent cannot reliably identify the underlying environment state. The POMDP formalism is very general and powerful, extending the application of MDPs to many realistic problems. They have become popular among researchers in Operations Research and Artificial Intelligence. In this work, we focus on estimating the state of the world of a Markov steady state system using the conditional distribution of observation for a given state of the world, and how fast it converges to steady state for different conditional distributions. Furthermore, we analyze the performance of Markov system in non- steady state conditions.

0.2 Partial Observability

Markov decision processes (MDPs) provide a mathematical framework for modeling decision making on a stochastic environment under the control of a decision maker. A POMDP is very similar to an MDP. It has a set of states, a set of actions, transitions and immediate rewards as in MDP. The consequence of actions on the state in a POMDP is also the same as MDP. However when it comes to the certainty of observation of the current state, they behave entirely different[2]. The MDP framework considers the environment as fully observable but POMDP framework considers it to be partially observable with some constraint of uncertainty.

In a POMDP we add a set of observations to the model. The observations are probabilistic and they give an idea of the state it is in. This observation function gives the probability of each observation for each state in the model.

It can be typically modelled as an agent interacting synchronously with a world. The agent takes as input the observation (with an uncertainty of state of the world) and generates as output actions, which themselves affect the state of the world, as shown in Figure 1. The policy (π) represents the mapping from observations to probability distributions over actions.

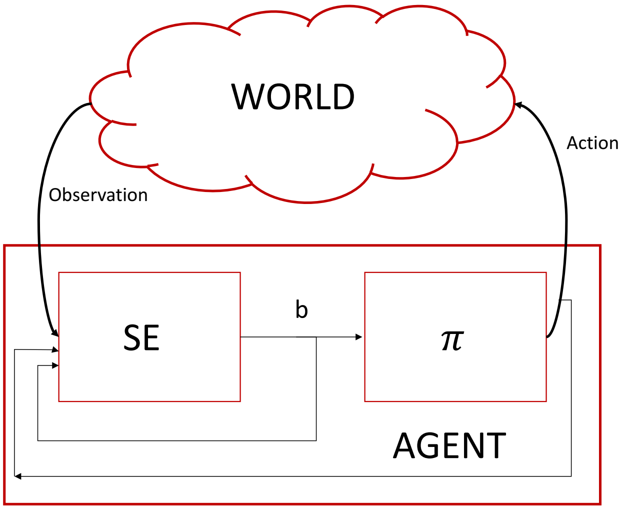

Figure 1: A POMDP agent can be decomposed into a state estimator (SE) and a policy (π)

0.3 POMDP Framework

A partially observable Markov decision process can be described as a tuple $(\chi, A, T, R, \Omega, Y)$

- χ is a finite set of states of the world
- A is a finite set of actions
- T: $\chi \times A \rightarrow \pi(\chi)$ is the state-transition function, giving for each world state and

agent action, a probability distribution over world states

- R: $\chi \times A \rightarrow R$ is the reward function, giving the expected immediate reward gained by the agent for taking each action in each state

- Ω is a finite set of observations the agent can experience of its world

- $Y : \chi \times A \rightarrow \pi(\Omega)$ is the observation function, which gives, for each action and resulting state, a probability distribution over possible observations.

The tuple (χ, A, T, R) represents a Markov decision process [5].

In POMP, the agent shown in Figure 1 is unable to observe the current state but makes an observation based on the action and resulting state. It's goal is to maximize expected discounted future reward. It makes observations and generates actions. It's previous experience is summarized in an internal belief state b. The component labeled SE is the state estimator which is responsible for updating the belief state based on the last action, the current observation, and the previous belief state. The component labeled π is the policy that is a function of the agent's belief state rather than the state of the world.

Probability distributions over states of the world are usually taken as the belief states. These distributions encode the agent's subjective probability about the state of the world and provide a basis for acting under uncertainty. They also comprise a sufficient statistic for the past history and initial belief state of the agent. i.e, given the agent's current belief state, no additional data about its past actions or observations would supply any further information about the current state of the world[5]. Thus the process over belief states is Markov, and that no additional data about the past would help to increase the agent's expected reward.

In this project, we analyze the robustness of the estimation of states in POMP using conditional distributions of states over observations. The analytical model of the problem is discussed in the next section.

0.4 Analytical Model

Let the transition probability matrix (Stochastic matrix) of the Markov chain be

$$P = [p_{ij}], \tag{1}$$

where p_{ij} is the probability of transition from state i to state j. It is expressed as

$$p_{ij} = P(x(t+1) = j/x(t) = i) \tag{2}$$

Here $x(t)$ be the state at time t where $x(t) \in \chi$. Assume $y(t)$ to be the observation made at time t and $\pi_0(.)$ be the initial distribution given by

$$\pi_0(x) = P(x(0) = x). \tag{3}$$

Let's denote the history of past measurements by

$$y^t = (y(0), y(1), ..., y(t)) \tag{4}$$

Our task is to compute $P(x(t) = x/y^t)$ for estimating the current state of the world. Let's define

$$\begin{aligned} \pi_{0/0}(x) &= P(x(0) = x/y(0)) \\ &= \frac{P((x(0) = x) \cap y(0))}{P(y(0))} \\ &= \frac{\pi_0(x)P(y(0)/x)}{\sum_{\overline{x}} \pi_0(\overline{x})P(y(0)/\overline{x})} \end{aligned} \tag{5}$$

for all $\overline{x} \in \chi$.

Further extending the definition for longer observation interval gives

$$\begin{aligned} \pi_{k+1/k+1}(x) &= P(x(k+1) = x/y(k+1)) \\ &= \frac{\sum_{\overline{x}} \pi_{k/k}(\overline{x})p_{\overline{x}x}P(y(k+1)/x)}{\sum_{\overline{x}}\sum_{\overline{x}} \pi_{k/k}(\overline{x})p_{\overline{x}x}P(y(k+1)/\overline{x})} \end{aligned} \tag{6}$$

where

$$p_{\overline{x}x} = P(x(k+1) = x/(x(k) = \overline{x}) \cap y^k) \tag{7}$$

Here $\pi_{k+1/k+1}(.)$ is the normalized conditional distribution of the state. It is equivalent to the Aposteriori probability distribution in Estimation theory. If there are only two states in the POMDP, the problem of state estimation essentially converges to a Bayesian estimation problem.

Let $\rho_{k+1/k+1}(.)$ be the unnormalized version of $\pi_{k+1/k+1}(.)$. It is also called hypostate or information state[4]. After introducing $\rho_{k+1/k+1}(.)$ into the system, Equation 5 is transformed to

$$\rho_{0/0}(x) = \pi_0(x)P(y(0)/x) \tag{8}$$

and Equation 6 gets changed to

$$\rho_{k+1/k+1}(x) = \sum_{\overline{x}} \rho_{k/k}(\overline{x})p_{\overline{x}x}P(y(k+1)/x) \tag{9}$$

Define the vector corresponding to $\rho_{k/k}(x)$ as

$$\rho_{k/k} = [\rho_{k/k}(1) \quad \rho_{k/k}(2).......] \tag{10}$$

We introduce an $N \times N$ matrix D which is a diagonal matrix with the diagonal entries as

$$D_{ii}(y) = P(y/x(i)). \tag{11}$$

where N represents the cardinality of the state space χ.

Thus Equation 8 and Equation 9 can be rewritten as

$$\rho_{0/0}(y^0) = \pi_0 D(y(0)). \tag{12}$$

and

$$\rho_{k+1/k+1}(y^{k+1}) = \rho_{k/k}(y^k)]PD(y(k+1)) \tag{13}$$

respectively.

0.5 Markov chain simulation and POMDP estimation - code specifics

In this section we explain the specifics of our code and explain how the POMDP estimate was arrived

1. Let π_0 be the actual starting state of the markov chain. Let there say 4 states in the markov chain

2. Take the interval $[0, 1]$ and divide it into 4 parts, each division point being the CDF $x \leq n$

3. Sample a uniform random variable

4. Check to which interval the sample lies in and that is the initial state of the markov chain

5. It is worth noting that if this process is repeated time and again, we get the required initial distribution

6. Once we get the current state, take the transition probability vector from the transition probability matrix for the current state and as was done for the previous step, we divide $[0, 1]$ according to the states, sample and get the next state

7. If this process is repeated, we get one run of the markov chain say of 500 steps

8. If these 500 steps are repeated, time and again for say 10000 times, we get the no of times the chain was in a particular state at a paticular step. we divide the no

of time in the state in a particular step by no of time the chain was in the step to get the occupation probabilities at that state

In order to do the POMDP estimation, we do the following

1. We set an estimate of the initial state say $\pi_0^{'}$ and then we sample the value
2. That value is the state of the system. But we observe only the noisy state of the system
3. If the states are $S = \{1, 2, 3, 4\}$ we get a 1 with probability $p_s = \{p_1, p_2, p_3, p_4\}$ and 0 with $1 - p_s$
4. We simulate a uniform random variable based on the p_i.Based on this observation, we construct the POMDP estimate using 13
5. We use this estimate as true distribution and continue the process

0.6 Problem statement

In the POMDP framework, for it to correctly estimate the state of the system (State transition probabilities), we need the initial distribution. In this project we look at the following

1. What is the error in the estimation due to improper initialization of the initial state?
2. Is it possible for POMDP to give a correct estimate of the state despite wrong initialization of the initial state? ie: π_0 .
3. What would be the error even if we initialize the POMDP correctly?

We study the properties by looking at the following two examples.

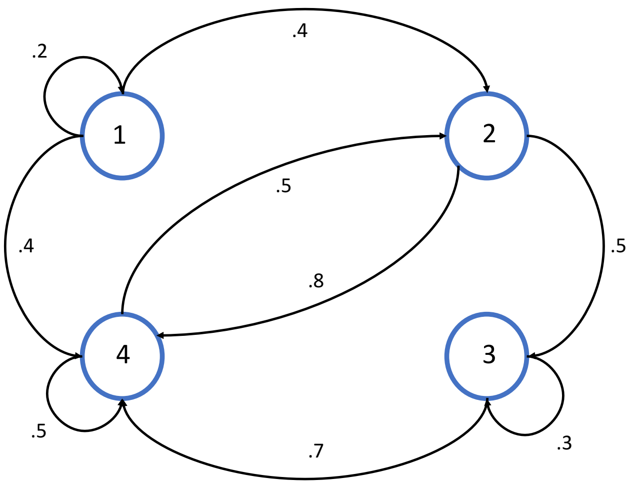

Figure 2: State Diagram for Markov Model 1

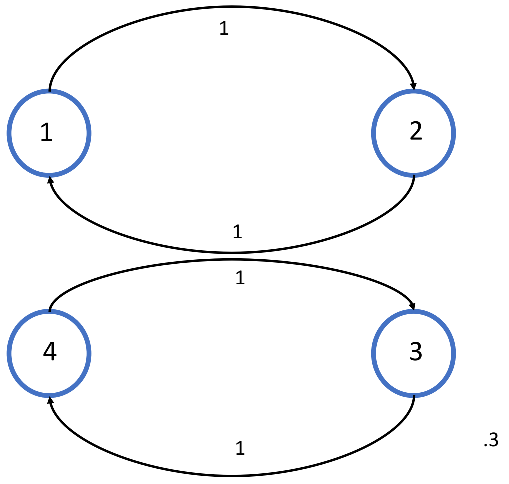

Figure 3: State Diagram for Markov Model 2

0.6.1 Model 1

The model 1 in Figure 2 consists of an irreducible Markov chain with 4 states. Depending on the state, we noisily observe either 0 or 1 with probability 1-p, p respectively which depends on the state. Let p_s denote this probability.

We have the following simulation results.

The transition matrix

$$\begin{bmatrix} 0.2 & 0.4 & 0 & 0.4 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.3 & 0.7 \\ 0.1 & 0.7 & 0 & 0.2 \end{bmatrix}$$

Let us start the simulation of the model with the distribution $p_s = [0.11110.22220.44440.2222]$ and let the actual distribution also be [0.11110.22220.44440.2222]. We estimate the Markov chain for 500 transitions and we take the L_2 norm between the actual probability distribution and the estimated probability distribution due to POMDP and we get the following

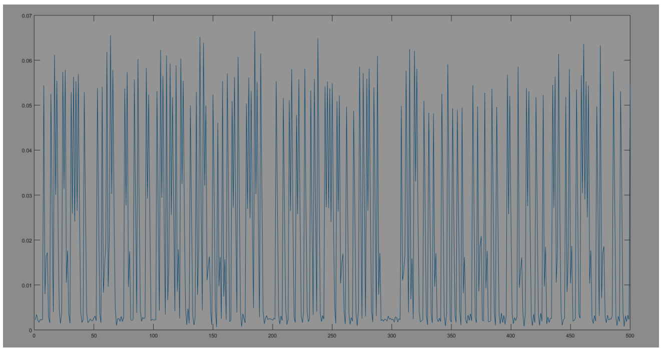

Figure 4: simulation 1

As seen in the figure, the norm difference has more or less a similar value of 0.02 on an average which is a very good estimate

We now repeat the simulation for various other scenarios

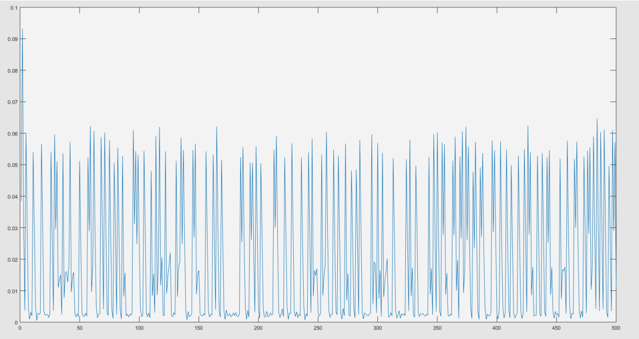

Figure 5: simulation 2

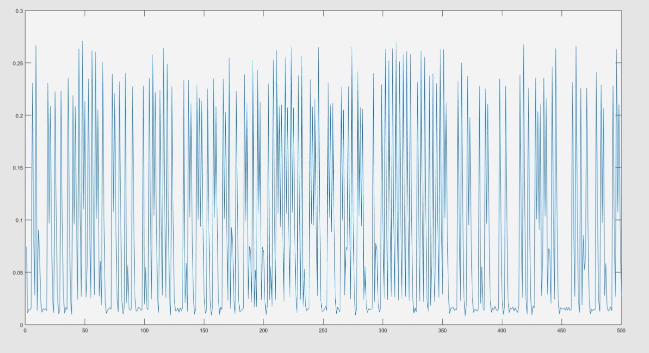

Figure 6: simulation 3

Simulation	Observation	initial distribution	POMDP distribution	Norm difference
S1	[0.2,0.4,0.2,0.3]	[0.1111,0.2222,0.4444,0.2222]	[0.1111,0.2222,0.4444,0.2222]	0.02
S2	[0.2,0.4,0.2,0.2]	[0.0519,0.3111,0.2222,0.4148]	[0.1111,0.2222,0.4444,0.2222]	0.0159
S3	[0.1,0.6,0.15,0.15]	[0.0519,0.3111,0.2222,0.4148]	[0.1111,0.2222,0.4444,0.2222]	0.0843
S4	[0.7,0.2,0.05,0.05]	[0.0519,0.3111,0.2222,0.4148]	[0.1111,0.2222,0.4444,0.2222]	0.0245

The simulations show that the convergence is quite fast and it remains so after each iteration step of the markov chain. On the other hand consider the second markov chain. The simulation results are as follows

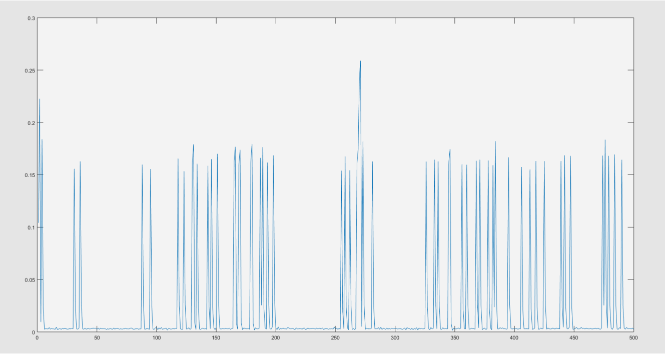

Figure 7: simulation 4

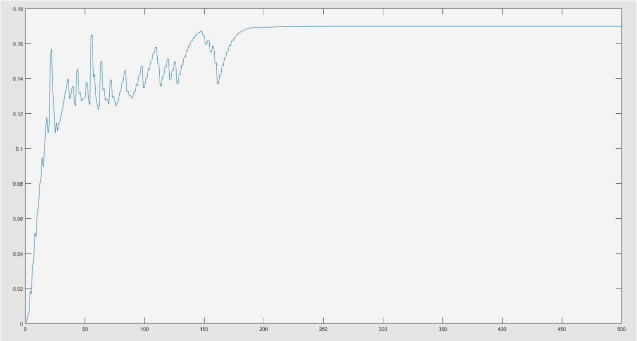

Figure 8: simulation 5

Simulation	Observation	initial distribution	POMDP distribution	Norm difference
S5	[0.2,0.4,0.2,0.3]	[0.1111,0.2222,0.4444,0.2222]	[0.1111,0.2222,0.4444,0.2222]	0.15
S6	[0.2,0.4,0.2,0.2]	[0.0519,0.3111,0.2222,0.4148]	[0.1111,0.2222,0.4444,0.2222]	0.2774
S7	[0.1,0.6,0.15,0.15]	[0.0519,0.3111,0.2222,0.4148]	[0.1111,0.2222,0.4444,0.2222]	0.2907
S8	[0.7,0.2,0.05,0.05]	[0.0519,0.3111,0.2222,0.4148]	[0.1111,0.2222,0.4444,0.2222]	1.0293

The markov chain considered was a reducible chain consisting of periodic components. The norm error between the actual distribution and the estimated distribution doesn't converge at all. It increases and stabilizes. Simulation S5 consists of starting the POMDP estimator in the same initial state as the original chain but the noise induced by D is so high that there is a norm error of 0.15

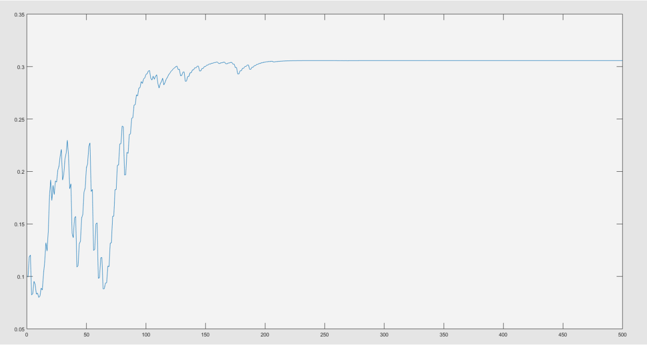

Figure 9: simulation 6

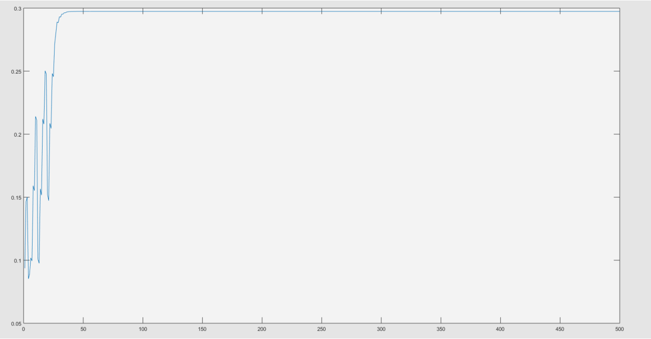

Figure 10: simulation 7

Hence by observing these markov chains we come to the following conclusions and pose the following questions

0.7 Conclusion

1. The dynamics of the markov chain affects POMDP estimation

2. It is seen from the second chain that the estimation is off by 0.15 even if the state of the estimator is initialized properly. Hence POMDPs are not good state estimators for certain types of markov chains. Possibly non irreducible ?

3. State estimation in the first model indicates that even if the state is initialized wrongly, the estimator converges very quickly to the actual state of the system

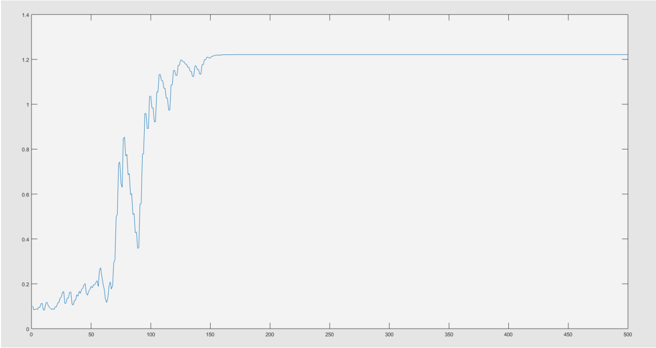

Figure 11: simulation 8

and this accuracy doesn't change as the markov chain progresses.It can be seen from the plot of the norm vs the time step of the markov chain for the first model that the norm error kind of oscillates around 0.02 to 0.05 on an average

0.7.1 Open problems from the observations

4. Is there a particular class of markov chains that make the POMDP more robust in terms of error estimation?

5. Is it possible that there is a class of markov chains that cannot be estimated properly by POMDPs (as in the second model simulated) ?

6. Could there be a better and more unified and robust estimator for state estimation ?

Bibliography

[1] Tony Cassandra. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley, New York, 1994.

[2] Tony Cassandra. *Tony Cassandra's POMDP page.* Brown University, 1999.

[3] Ronald A. Howard. *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, 1960.

[4] P. R. Kumar. *Stchastic Systems.* ECEN 755 Lecture Notes, 2018.

[5] Michael L. Littman Leslie Pack Kaelbling and Anthony R. Cassandra. *Planning and Acting in Partially Observable Stochastic Domains.* Artificial Intelligence, Vol. 101, 1998.