Improvise BSS with POMDP

Jia Lin Hau, Sepideh Koohfar, M.Rasel Mahmud, Xihong Su

Fall 2019

1 Introduction

The model we revisit in this paper is, Bayesian State Space (BSS) approach with Partially observable Batch Reinforcement learning. We introduced a RL algorithm that is more applicable to real world problem when (1) The states is not fully observable. (2) There is only small dataset available. (3) Does not interact with the simulator[2][6]. In section 2, we will explain the application and motivation of this approach. Section 3, we quickly explain an instance of Partially Observed Markov Decision Processes (POMDP). In section 4, we will Implement the BSS technique with POMDP on Cart Pole.

2 Motivation

Different types of pests causing a huge loss in the fruit/crop industry. Only one kind of pest SWD causes $1.28 billion crop loss annually in the USA[[4]. Growers need to spray insecticide in an efficient manner to solve the issue. The existing approach to spray insecticide is calendar-based which means growers spray insecticide in every few days after without monitoring whether they need to spray or not. Normally growers do not monitor as it is very costly and even if they place traps in many places, those do not provide perfect observation of the number of pests. So, the growers usually spray a lot more than they actually need. For these reasons, crop/fruit industry is searching for an efficient strategy to spray insecticide. So, our project goal was to develop a strategy to reduce the unnecessary use of insecticides in fruit orchards.

We found two previous works closely related to our project. One was based on the blueberry farm in Maine published in July 2019. Another one was based on 17 blueberry farms in New York and published in 2016. Both the model was based on Bayesian State Space (BSS) approach[4][10].

The existing solutions mentioned that POMDP should give better results. Because, POMDP can capture dynamically Bayesian updating of both the parameters of the models and the number of pests during the management process. But BSS can't do that. Also, existing BSS frameworks use a set of predefined strategies that are functions of trap count to rank the performance. But

POMDP can take optimal control decision in each period and provide fully optimal strategy. So, POMDP should work better.

3 POMDP in Grid World

For the maze game, the agent's grid world is (3,4) hardcoded matrices. The agent has no information at all about its position in non-terminal states, and it knows only when it is in a terminal state because the game ends. One terminal state has reward -10000, and the other one has reward 1000. The goal is to reach the best possible terminal state. In addition, the agent has 9 non-terminal states and one invalid state(because of the wall). The initial belief state probability of each position is 1/9(except terminal states and invalid state). The agent can take actions "Up", "Down", "Left", and "Right". For each action, the agent has 0.7 probability to reach the intended direction, and has 0.15 probability to move the right angle of the intended direction, and has 0.15 probability to move the left angle of the intended direction. Observations include "two walls", "three walls", "one wall" and "the end". Because of the noise effect, the agent has 0.8 probability to see the correct observations and 0.2 probability to see wrong observations. When the agent is in a terminal state, it can see "the end". We update belief state distribution based on formula in[9]. The experiments are based on Osman's work [8]. We create a list of random actions and a list of random observations. The inputs include initial belief state distribution, list of random actions, and list of random observations. The results show how many steps the agent move and where the agent could be with some probability.

4 Cart Pole

This is a domain from open AI gym where the goal is to balance the pole on a cart. With either action Left or Right. If the pole Angle exceed some angle or the cart go out of the window we will consider lose (Random action last 25 time steps on average), otherwise each time-step will get a reward of 1. We let Pole withstand specific time-step (500) considered win because it can eventually last forever[7].

4.1 Overview

(Section 4.2) We will first introduce the naive approach to solve this particular domain, we will explain why we use this (solved) domain for our purposes and the specific the modification we made to make cart pole relate to the real application. (Section 4.3) We will use Multiple Linear Regression (MLR) to solve this problem. (Section 4.4) We will then use Bayesian inference MCMC to generate samples and improve our result. (Section 4.5) We will hide the most crucial variable (Pole Angle) of the domain to decision maker and introduce (POMDP) to solve this problem.

4.2 Naive Approach

There is a very naive approach to cartpole domain[5]. There exist 4 variables (X) in this domain, include cart position, cart velocity, pole angle, and pole velocity. Where the user interact with the program and then solve for a = b1 x1 + b2 x2 + b3 x3 + b4 x4 = BX, If a>0 pick action 0, else pick action 1. Given a randomly select value for B. After N iterations choose B that gives the highest reward among all the iteration. One could use gradient descent or some similar strategy to obtain an optimal solution.

However, in this paper the policy maker can only interact once with the domain when they have a policy and use the domain to evaluate the performance of their policy. So technically, while coming up with a policy from the dataset, the policy maker could not interact with the domain. Another complication we add to the problem is where we only have 50 sets of data of randomly played cartpole game, 25 steps on average for each set of data.

In section 4.4, we will use our Bayesian Data generator on this approach and show that with the Bayesian Data generator, we could generate the data given from the cartpole simulator.

4.3 Multiple Linear Regression (MLR)

In this section we suggest to use supervise learning to understand the distribution for all dependent and independent variables in the dataset. We will then use a good supervise learning from this section and pass into (section4.45) STAN to generate more possible data points.

Note that this problem there exist a symmetric instance of the cartpole for every states where we multiplied all the variables by -1 and swap the value for action 0 and 1. By doing so, we manually generate the symmetric data. We now use the data that we have and apply MLR. We now split the dataset into 2 datasets, each for an action. For this specific problem, we do first order MLR to approximate the change in all the variables given action.

For example: $lm(\Delta PoleAngle \sim CartPosition + CartVelocity + PoleAngle + PoleVelocity, data = action_i)$

We obtain all linear fits with a Adjusted R-squared ≥ 0.972. The high R-squared might not be unreliable or unobtainable sometimes. We will use the same MLR with STAN (Section 4.4) instead of stats package.

4.4 Bayesian Data Generator (STAN)

From section, 4.3 we fit 4 different MLRs, each for a variable. In this section, we do the same with STAN, a probabilistic programming[1]. To verify this technique actually work, We mimic the procedure in 4.2 with this generated data with 100 different Bs 10 times each and pick the best B that yield the maximin reward, robust decision [3]. Then, we will use the best B from the STAN and apply to the simulator, the simulation reward is expect to be better than the maximin reward obtained by the STAN.

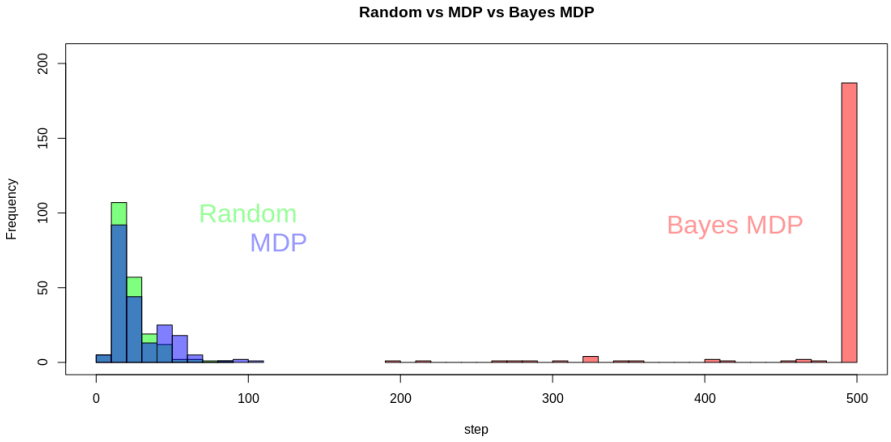
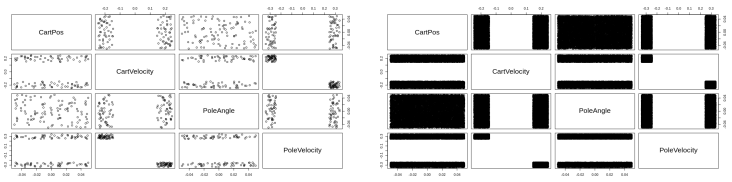


Figure 1: Reward Comparison Plot

For general application purposes we will use MDP to solve this problem. Note that, there exist infinitely many states for each variable because it is continuous. Therefore, we will need to use state aggregation to combined it to finite value of states. For each variable we split it into 10 different states, we split states by quantile, which outperform equal range state aggregation. We will end up with 10^4 states from 4 variables. For each state action pair we use the STAN output to generate 100 possible data to create the transition probability. In term of reward, we set states with angle in the range that we have seen achieve reward 1, otherwise zero. Same for the position of the cart. We also used a 0.95 discount. By doing so, the cart pole will try to maintain to stay at position and angle that it has seen but not result losing.

To study different technique of solving MDP, we use Value Iteration, Policy Iteration and Linear Programming with Gurobi, to solve this MDP and they all gives us a similar result. By implementing BSS with MDP it outperform the naive MDP. An initial distribution is created by populate the possible initial distribution from the dataset.



(a) Data Initial Distribution (b) Populated Distribution

Figure 2: Generate Initial Distribution.

4.5 POMDP implementation

The formulation of POMDP is similar to MDP, there are a couple more terms for POMDP compare to MDP. In this POMDP, we hid the variable Pole Angle, which is the most crucial variable in this domain. Our tuples for POMDP are defined as: * (generated from MDP in 4.4)

- Discount*
- States(Pole Angle,Pole Velocity,Cart Velocity) = 1:1000
- Observations(Pole Velocity,Cart Velocity) = 1:100
- Initial Start Distribution*
- Transition Probability*
- Observation Probability = given observation the probability that it is in a specific states, generated with STAN MLR same technique explained in 4.3
- Reward*

The formulation of POMDP we currently have does not work. I suspect there exist a bug in the formulation of observation probability. Therefore, more work is required to get this to work.

5 Future Work

We would like to get the POMDP to work and give us a promising result on Cartpole which is our baseline domain. If it does, we would love to use the same technique and apply it to the SWD dataset to manage invasive species.

References

[1] Jeffrey B. Arnold. bayesian_notes. https://jrnold.github.io/bayesian_notes/introduction-to-stan-and-linear-regression.html. Accessed on 2019-12-11.

[2] Patrick Dallaire, Camille Besse, Stephane Ross, and Brahim Chaib-Draa. Bayesian reinforcement learning in continuous POMDPs with Gaussian processes. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pages 2604–2609, 2009.

[3] Michael Doumpos. Robustness Analysis in Decision Aiding , Optimization , Associate Series Editor. 2016.

[4] Xiao Fan, Miguel Gomez, and Shadi Atallah. Optimal Monitoring and Controlling of Invasive Species: The Case of Spotted Wing Drosophila in the United States. *Agricultural and Applied Economics Annual Meeting*, (March 2018), 2016.

[5] Kevin Frans. Simple reinforcement learning methods to learn cartpole. http://kvfrans.com/simple-algoritms-for-solving-cartpole/, Jul 2016. Accessed on 2019-12-11.

[6] David M. Kling, James N. Sanchirico, and Paul L. Fackler. Optimal monitoring and control under state uncertainty: Application to lionfish management. *Journal of Environmental Economics and Management*, 84:223–245, 2017.

[7] OpenAi. Cartpole-v0. https://gym.openai.com/envs/CartPole-v0/, 2015. Accessed on 2019-12-11.

[8] osmanhajiyev. Pomdp. https://github.com/osmanhajiyev/POMDP/blob/master/POMDP.java, Oct 2017. Accessed on 2019-12-11.

[9] Bogdan M. Wilamowski and J. David Irwin. Intelligent Systems, 2016.

[10] D Adeline Yeh, Miguel I Gómez, Francis Drummond, Xiaoli Fan, and C.-Y. Cynthia Lin Lawell. A Farm-level Bioeconomic Model of Invasive Species Management: The Case of Spotted Wing Drosophila in Maine. 2019.