Bayesian MDP solved Cartpole with limited data

Jia Lin Hau

Fall 2019

1 Introduction

The model we evaluate in this paper is, Bayesian Inference (MCMC) Partially observable Batch Reinforcement learning. We introduced a RL algorithm that is more applicable to real world problem where (1) The states is not fully observable. (2) There is only small dataset available. (3) Does not interact with the simulator. (Section 3) We will first introduce the CartPole domain, why we use this (solved) domain for our purposes, and the specific modification we made to make it way harder. (Section 3.3) We will use Multiple Linear Regression (MLR) to solve this problem. (Section 3.4) We will use Bayesian inference MCMC to generate samples and improve our result. We showed that with only one batch (10 trajectories) of randomized off-policy datasets, our algorithms was able to completely solved the Cartpole MDP.

2 Motivation

1. Impractical to have big dataset.
2. Most of the problem does not have fully observable state space.
3. Generalized RL method that could be used in more domain.

While big data is becoming a hot topic, small data is still remain obscure. In agriculture, natural disaster, health care, and natural resources having big dataset is impractical. Nevertheless, making a good decision is important. Our work is used to solve an issue of batch reinforcement learning where data might be insufficient to compute a good policy. We propose a robust reinforcement learning approach that can compute good solutions even when the models or rewards are not known precisely. We use Bayesian models to capture prior knowledge of our feature. We propose to use (STAN) a full Bayesian statistical inference with MCMC sampling. This procedure works well for domain that has low action-space and big state-space.

3 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 (200) considered win [4].

3.1 Overview

(Section 3.2) 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 3.3) We will use Multiple Linear Regression (MLR) to solve this problem. (Section 3.4) We will then use Bayesian inference MCMC to generate samples and improve our result. (Section 3.5) We will hide the most crucial variable (Pole Angle) of the domain to decision maker and introduce (POMDP) to solve this problem.

3.2 Simple Linear Regression

There is a very naive approach to cartpole domain[3]. 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 10 sets of datas of randomly played cartpole game, each with an average of 25 steps.

In section 3.4, 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.

3.3 Multi-variate 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 (section3.4) 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 3.4) instead of stats package.

3.4 Bayesian Data Generator (STAN)

From section, 3.3 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 3.2 with this generated data with 100 different Bs 10 times each and pick the best B that yield the maximin reward, robust decision [2]. 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.

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.

4 Conclusion

We can see from (Figure 1) that the Bayesian Model-based MDP outperforms the standard MDP. With only limited data and unable to interact with the domain, we can use Bayesian supervised model-based reinforcement learning techniques to obtain the optimal solution for this problem.

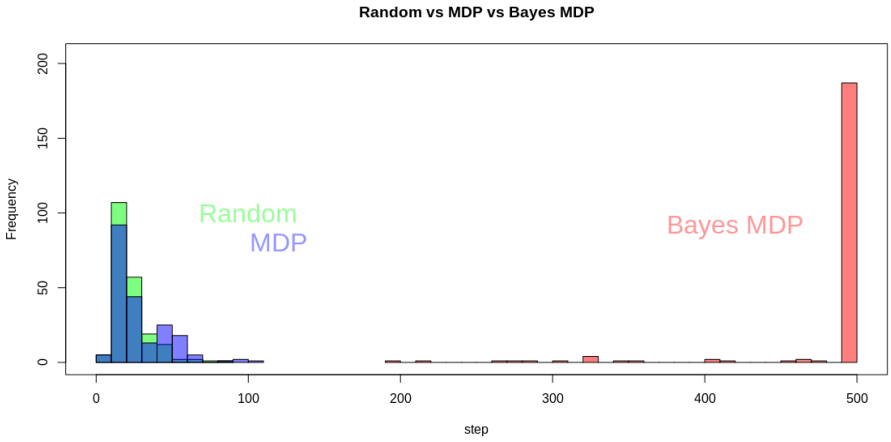
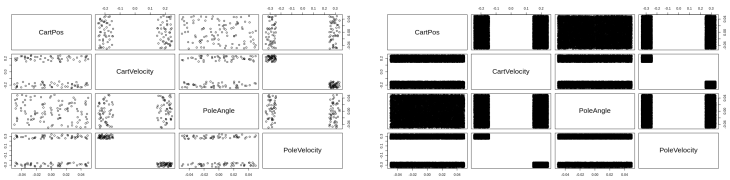


Figure 1: Reward Comparison Plot



(a) Data Initial Distribution (b) Populated Distribution

Figure 2: Generate Initial Distribution.

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

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

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