PERFORMANCE COMPARISON OF VALUE ITERATION AND SARASA FOR THE BEER PONG GAME

Nishith Burman

Northeastern University

ABSTRACT

The objective of this project is to discuss value iteration and SARSA in detail. Here I discuss how does these two algorithms work, how to get the optimal policy using these algorithms, why is it better than policy iteration and also the cases when this algorithm won't be effective. We explore these algorithms by implementing it on a simple real-world problem of throwing a ball at its target from any random point of the room. Finally, we present the optimal policy calculated by these algorithms and compare them.

1. INTRODUCTION

Markov Decision Processes (MDP) is a commonly used formalism for modeling systems that use both probabilistic and non-deterministic behaviors. These are generalizations of discrete-time Markov chains for which non-determinism is forbidden. MDPs have acquired an even greater gain of interest since the development of quantitative verification of systems, which in particular may take into account probabilistic aspects for a deep study of model checking techniques, in particular for probabilistic systems. Automated verification techniques have been extensively studied to handle such probabilistic models, leading to various tools like the PRISM probabilistic model checker.

There are various different methods and algorithms (e.g. Linear Programming, Policy Iteration) that can be used for the model-checking of MDPs and Markov. But there are some problems associated with these techniques. The first simple, yet intriguing, the problem lies in the computation of minimum and maximum probabilities to reach a target set of states of an MDP. Exact polynomial-time methods, like linear programming,exist to compute those probabilities, but they seem unable to scale on large systems. Nonetheless, they are based on the fact that these probabilities are indeed fixpoints of some operators. Usually, numerical approximate methods are rather used in practice, the most used one being value iteration. The algorithm consists in asymptotically reaching the previous fixpoints by iterating the operators.

Now function approximation techniques like policy iteration can also be used for this task. But policy iteration has a major drawback i.e. It involves policy evaluation in each of its steps. which may itself be a protracted iterative computation requiring multiple sweeps through the state set. Now, this issue can be tackled by using a lot of different methods and there are various different ways to truncate the policy evaluation step of policy iteration without losing the convergence guarantee. Of all those ways Value Iteration is the best technique in which policy evaluation is stopped just after one sweep.

Figure 1 above shows the game of Beer pong where the objective is to send the balls into the cup. Everyone gets fixed no. of balls and those who send the balls into the cup maximum no. of times win. Here in our setup, additional leverage has been provided to the agent

2. PROPOSED WORK

2.1. PREVIOUS WORK

A lot of related work has been carried out and various different algorithm has been implemented for this task previously to calculate the optimal policy and also for improving this algorithm further. For instance, S. Osborne used various different kinds of algorithms like Temporal difference algorithm (TD(0)) and Q- learning algorithm to solve this task. In his work, he was basically checking whether model-free learning is better than model-based learning.

2.2. OUR APPROACH

Through this project, I am not trying anything new like improving some aspect of the value iteration model neither am I modifying any aspect of the algorithm. Rather than that my main goal here is to explore each and every aspect of value iteration algorithm and SARSA and find out the best action sets for both of them. Not only I am using the value iteration and SARSA algorithm to find the best possible policy from any given state but we are also discussing here what is the advantage of this algorithm over the normal policy iteration algorithm.

we are discussing the scenario in which the value iteration algorithm won't be really effective. Many of the things that have been done and explored here are already taught to us during lectures but the only extra thing that I am introducing here is the visualization of the best action from every possible state in our space i.e whether to throw the ball or to adjust the position of the agent. The visualization also shows the direction in which the agent needs to move in order to reach a better location for throwing the ball. Finally, we are comparing which of the above-mentioned algorithm is performing for this particular task based on their performance for a specific no. of episodes.

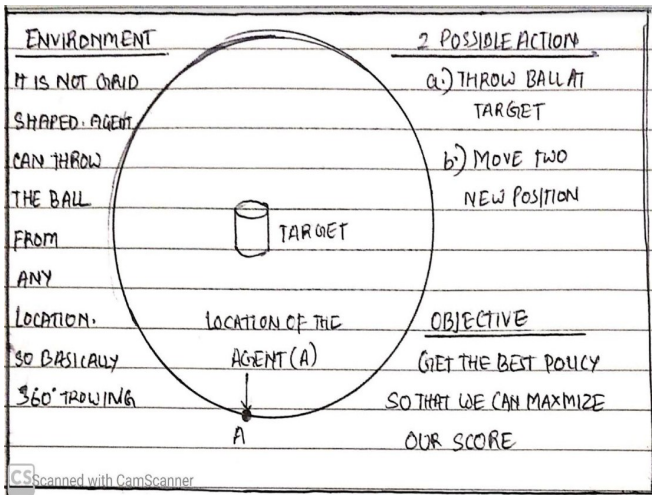

Figure 2. above shows the visualization of the basic environment . The environment is basically 3-D version of the actual room. Target is at the center of the room and the agent is situated in location A of the room . .

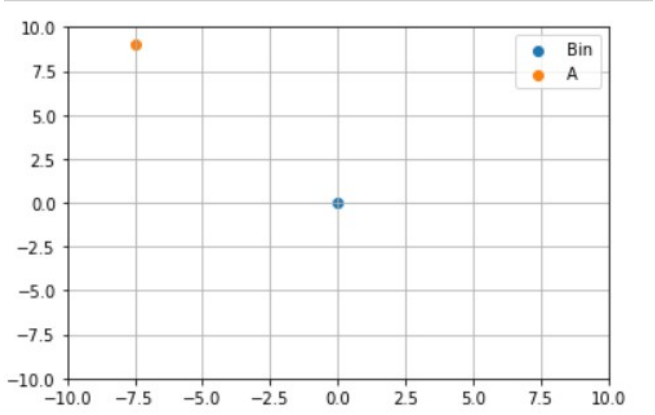

Figure 3 above show the mapping of the environment in the 2-D space. Here The blue dot is representing the location of the target and the orange dot is representing the location of our agent. Clearly, we can see that there are total of 100 different states from the above representation

3. MODEL FRAMEWORKS

3.1. MATHEMATICAL CALCULATION REQUIRED TO SOLVE THE PROBLEM

Here our objective is to throw the ball into the cup. But the distance and direction of the throw are the two factors that contribute to a successful throw. Better are these two parameters better will be the probability of successful throws.

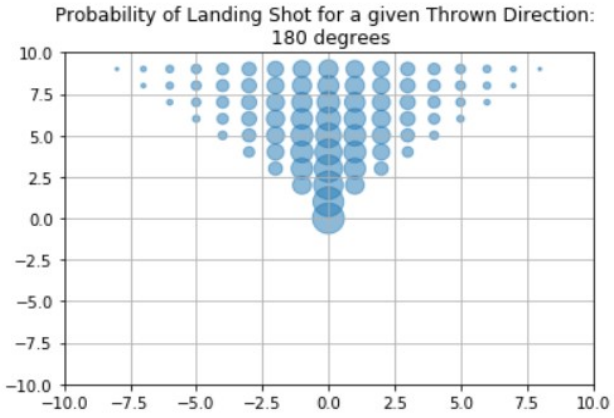

Figure 4. above showing the probability of hitting the target from different locations of the room. Here it indicates just the east side of the room.

3.2 CALCULATION OF DISTANCE

There are different methods to calculate the distance of the current state or position of agent from the bin. For instance, we can use Manhattan distance or Euclidian distance measure. But we will be using Euclidean distance measure for our calculation purposes. Now Euclidean distance can be calculated by the formula

Distance(State, Cup) = sqrt((state(X), Cup(X))^2 + (state(Y), Cup(Y))^2)

Now we also need to normalize this distance in order to calculate actual proximity of the agent from the cup which can be done by using the formulae

Norm_Score(state, cup) = 1-Distance(state, cup)/ Max_Distance(state, cup)

3.3 CALCULATION OF DIRECTION

Apart from the distance direction of the throw is another factor that affects the probability of hitting the target. The direction of throw can be calculated by using simple trigonometric formula written below

Direction(state, cup)= tan^-1(origin(x)–state(x)) /(origin(x)–state(x))

But if the agent throws the ball at a different angle then in that case, we will also need to get the one more angle to verify how good agents chosen direction was. This angle is represented by the formula below-

Dir_Score = T_direction-|T_direction-Direction| /T_direction

3.4 PROBABILITY OF HITTING THE TARGET CALCULATION

The probability of successfully hitting the target can be calculated by is the joint probability of Norm_Score and Dir_Score (Here both the event are independent) –

P_Success(state,Action)=Norm_Score*Dir_Score

4. FINDING THE OPTIMAL POLICIES FOR THE ENVIRONMENT

Now for our problem, we are having everything that we need to know to find better action i.e. state values, action and transition probabilities. Thus, basically, we are performing model-based learning. If I represent my problem in grid world like format then I get a 10*10 matrix So basically there are 100 states in for our problem and from each state, we can perform two actions 1.) Throw 2.) Adjust location. Now we are

updating our values based on value iteration. Mathematically it can be represented by the expression shown below

$$v_{k+1}(s) \doteq \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a]$$
$$= \max_a \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_k(s')\Big],$$

Similarly, for updating the action values in SARSA we can use the equation represented by the expression shown below

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big]$$

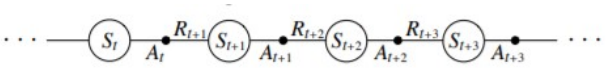

Figure 5 is portraying the SARSA updated diagram. Here the current state, action update is based on the next state and action.

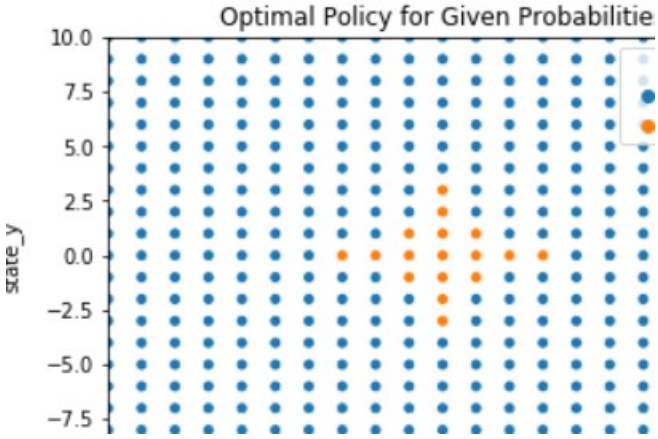

Figure 6 above depicts the action that is needed to be taken from all the possible sets of 100 different states. Here the blue dots are indicating that it is better to adjust location while the orange dots are indicating it better to throw from that position

5. MODEL EVALUATION PARAMETER

We are not actually using any kind of machine learning or deep learning algorithm. So, I can't use the normal model evaluation metrics for this problem. But what I can do here is show a plot of Q value with respect to the updates.

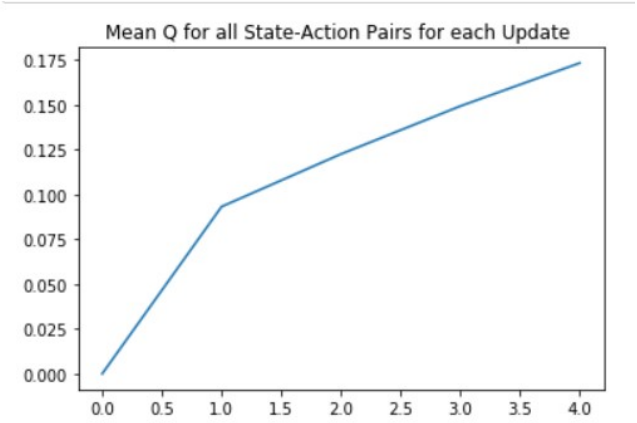

Figure 7 here shows the average Q values of the state action pair during each updates in the initial stages when the agent hasn't taken a lot of actions within the environment.

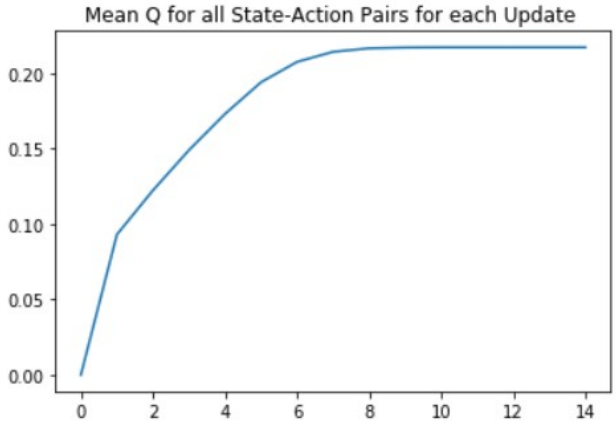

Figure 8 here is showing the Q values of the state action pair after each update(This is when the agent has found the optimal policy i.e. Agent knows the best action from each individual states). It is evident from both the plot that the average Q values for the 2nd plot is greater than the average Q values of the 1st plot. This is due to the fact that we have found the optimal action for each state and thus getting better rewards and better Q values.

The above two figure are for value iteration. Now let see the state's best action values based on the

episode. For our problem, we are running the SARSA algorithm for 1000 episodes. Based on the result obtained from the first 1000 episode we can conclude that value iteration performs slightly better until the first 1000 episode

Figure 9 is showing the best action values for each episode until 1000 episodes. We are getting values between 0 to 0.1. So basically, these values is slightly less than the values obtained in value iteration but of course, these values can change if we change the model parameters like alpha or if we increase the no. of episodes to 10000

6. MERITS AND DEMERITS OF THE VALUE ITERATION

6.1 MERITS OF VALUE ITERATION ALGORITHM

One of the major advantages of using the value iteration instead of typical policy iteration is that for the former case we don't need to do policy evaluation during each sweep and thus in value iteration, computational complexity is reduced greatly.

6.2 DEMERITS OF VALUE ITERATION ALGORITHM

Now whatever we have done until now, we have information about each and every model parameter, for instance, we had the knowledge of state values, Probability of taking an action and even the state transition probabilities. But in real-world setting very often we do not have all required information. For an instance in many situations we won't be having the state transition probability for the purpose of updating the values. So, in such kind of situations value iteration won't be that much effective for getting the optimal policy and for finding optimal action from each state. Of course, to tackle the above issue we can use alternative methods like Q-learning, SARSA or Dyna Q for the purpose of finding the optimal policies and optimal action from each state.

6. MERITS OF SARSA ALGORITHM

There are two basic advantages of using SARSA over Value iteration algorithm. First and Foremost, important advantages is that it is a model-free learning method and does not need complete information of each and every parameter of the environment i.e. basically this algorithm can update the state values even if transition probabilities are not available. Secondly, updates in SARSA are just based on the next state action – value pair and doesn't consider every state for update thus this algorithm is considerably faster.

7. CONCLUSION

I have used a simple task for comparing the value iteration algorithm and SARSA algorithm and to find the optimal actions from each states and optimal policy from any given point. I have also demonstrated the underlying concept behind the value iteration algorithm and the use of Euclidean distance and trigonometric functions for calculating the probability of hitting the target. Further, I discussed the merits and demerits of this value iteration algorithm. Apart from this I also discussed why this algorithm won't work in case of
Model-free environment and also discussed in detail SARSA which is a model-free learning algorithm and can be used in the case where there is lack of Transition probabilities. I also presented different Visualization to clearly show the performance of both the model. I found for this problem Value Iteration is performing better by a very small margin (Although this can change

as we run SARSA algorithm just for 1000 episodes).

7. FUTURE WORK

A variety of different algorithms have already been applied for this task. For Instance, temporal difference and Q-learning have already been applied to this problem. Now we have applied Value iteration and SARSA. But there are still two different possibilities that can be tried in the future. So, we can try implementation of the Dyna-Q algorithm and Dyna-Q+ algorithm. We can also try to implement some Q-learning in combination with Neural Network (Basically Deep Q learning) and then one can analyze the performance of each algorithm.

8. REFERENCES

1.) Finite time bound for fitted value iteration R Munoz, C Szepesvari, Journal of Machine Learning Research 2008

2.) Topological Value Iteration algorithm for Marakov Decision Process by P Dai, J Goldsmith 2007, IJCAI

3.) Reachability in MDP's : Refining convergence of the Value Iteration model by S Haddad, B Monmege, Springer 2014

4.) Bakward Q-Learning : The combination of SARSA and backward Q-learning by YH Wang, THS Li, CJ Lin, Elsevier, 2013

5.) Reinforcement Learning , Richards S. Sutton and Andrew G. Barto

6.) Applying model based and model free methods and evaluating parameters in detail by S. Osborne

7.) Simple Reinforcement Learning with Tensorflow by Arthur Juliani

8.) Deep Reinforcement learning demysitifed : Policy Iteration, Value Iteration and Q-Learning

9.) Introduction to various Reinforcement Learning Algorithms. Part1(Q-Learning, SARSA, DQN, DDPG) by Kung-Hsiang, Huang.