Homework 3 – Reinforcement Learning

Dynamic programming is a method by which we solve complex problems by breaking them down into subproblems. Then solving those subproblems and combine solutions to the subproblems to solve the overall problem

The premise of the problem:

In this homework set, we are to compute the policy iteration and value iteration algorithms for a Gambling game, as explained in the class. We calculate the policy iteration algorithm for the case when the probability of heads is 0.9. We compute the value iteration algorithm for the case when the probability of heads is 0.4

Policy iteration algorithm :

Policy iteration algorithm is done three main steps of initialization, policy evaluation and lastly the policy improvement.

We do the iteration and the policy evaluation in a similar as in homework 2. Here we choose a policy at random. The policy selects an action to bet randomly from the available actions, which in turn is based on the states. Next, I will briefly go over the policy evaluation code. So, we then define a matrix that allows us to have a defined way as to at which what state which action has greater or lesser possibilities of being taken. In short, the matrix, rows are stated, and the columns are actions. Now next we make a while condition to calculate the value of delta, I will get to it in a later part of the explanation. We then define an array of the values to be equal to zero and then begin to loop for states and the actions. The reward matrix is based on the action being selected at that moment. Using the policy matrix previously defined and the loops of the state and the actions we can easily write the probability of the action being taken under that policy. Next, we define the various cases wherein the Bellman's update / backup formula. This helps us use an array version wherein we update the new values onto a vector. This is then continuously done until for that policy all the states and actions have been exhausted. Also, in the bellman's update formula, we define both the probabilities for head and tails and use this to update our value parameter. Once we have done for all the states and actions for a policy, we subtract this from the older value and find its max. upon comparing this with our previously mentioned delta. Delta is our convergence parameter and helps us converge the code itself.

Once we have done policy evaluation, we need to do the policy improvement in order to yield a better policy, which we can then use to compute another policy and then yield an even better one. Basically, the way of finding an optimal policy is known as policy iteration. Now in the policy improvement we pass in this policy that we obtained by using the policy evaluation algorithm, we first assume that this policy is stable, and we initialize a new matrix of policy as well which is an array. We next loop for the states and actions and calculate the various values here as well using the various conditional statements. After this, we append the various values to a matrix and try and find the maximum or the greedy among them. Then with this greedy matrix, we construct a new policy. Now we compare if the new policy is equal to the old policy and if this is the case, we update a flag, else we say it's not stable.

Now if this is an unstable policy that is isn't a better policy, we apply this obtained value to calculate a new policy evaluation and then repeat the policy iteration again .

Output for Policy Iteration

```
policy_used =

     1     0     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     1     0


Value_obtained =

   13.9268   14.5853   13.7696   12.7901   11.7924   10.7926    9.7927    8.7927    7.7927
```

Value iteration :

So, a drawback of policy iteration is that as we see that each of its iterations involves policy evaluation and this involve multiple computations and sweeps being required . This is made much less computationally challenging in value iteration algorithm . Firstly, we define a value delta which is used to determine the accuracy for the estimation.

Here we choose a policy at random. The policy selects an action to bet randomly from the available actions, which in turn is based on the states. Now next we make us of a while condition to calculate the value of delta, I will get to it in a later part of the explanation. We then define an array of the values to be equal to zero and then begin to loop for states and the actions. The reward matrix is based on the action being selected at that moment. Using the policy matrix previously defined and the loops of the state and the actions we can easily write the probability of the action being taken under that policy. Next, we define the various cases wherein the Bellman's update / backup formula. This helps us use an array version wherein we update the new values onto a vector. This is then continuously done until for that policy all the states and actions have been exhausted. Also, in the bellman's update formula, we define both the probabilities for head and tails and use this to update our value parameter. Once we have done for all the states and actions for a policy, we subtract this from the older value and find its max. upon comparing this with our previously mentioned delta. Delta is our convergence parameter and helps us converge the code itself.

We next do a policy improvement of the obtained policy as similar to the policy iteration algorithm , but the difference here is we don't have to wait for the value to converge and we can do the improvement with a small evaluation and the value iteration policy is able to converge faster .

Output for Value Iteration

```
policy =

     1     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0
     0     0     0     1     0     0     0     0     0
     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     1     0     0     0
     0     0     1     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0


V_USED =

   -0.5520   -0.8800   -1.0800   -1.2000   -1.0000   -1.2000   -1.3200   -1.1200   -0.8720
```

Formula being used :

To implement the value functions for the policy, we make use of the following method: V(s)=

$$= \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a) \Big[r + \gamma v_{\pi}(s') \Big], \quad \text{for all } s \in \mathcal{S},$$