למידה במערכות דינמיות אביב תש"פ

תרגיל בית 3

מגישים:

יאיר נחום 034462796

דר ערבה 205874951

* 1. It is easy to see that the optimal policy must be to reach state n as fast as we can since the reward is 0 on every other state. Thus, the policy is to go right on every stage. In stage n we have no other option than to go back to 0.
  2. For the fixed policy we can apply policy evaluation by solving |S| equations (the same as calculating inverse matrix ). Also, we notice that in this case the MDP is deterministic once we know the action by the policy:

We can solve these equations by noticing the connection between the first stage and the last stage:

* 1. According to total probability on the actions, we can calculate the transitions matrix as follows:  
     for each state other than n, the policy is stochastic with 0.5 chance for every action.  
     In order to calculate the stationary distribution of the states, one needs to solve the vector equation:

Or per state probability:  
According to the formula above for the transition probabilities we have n equations and also the demand:

We note that the dynamics are deterministic once the action is selected, thus, the transition probability is actually the policy probability.

Again, by applying the recursion of the equations and using the normalization to a probability vector we get:

* 1. FPVI specifies

In matrix description   
In our case, since we started with then we get:   
The transition matrix can be calculated using:

On state 1 we have only one action possible, so we can only move to state 2.  
On other states other than n, we select an action randomly with probability 0.5 for each action and then continue the dynamics deterministically.  
Thus:  
The same for going from i state to 1.  
On the n state we have no option than going back to 1 state.  
When:  
Therefore, If = 0, then:

From FPVI convergence Proposition 5.2, we get that   
BTW, can be calculated directly from as we proved in Lemma 5.2 in the lecture.

* 1. We can apply recursive expectation calculation. First, let’s observe what happens in simple cases like n=2,3,4:  
     Let’s denote in the time (expectation of it) to reach state n from state k.  
     We want to find   
       
     In case of n=2:

As there is only one possible transition to state 2.

In case of n=3:  
Another way to calculate it is by explicit total expectation:

When in our case, the p denotes the probability to move from state i to state i+1.

In case of n=4:

In the general case:  
   
for

* 1. In order to go through all state actions pairs, we will define a policy such that at the first time we encounter some state i we get back to 1 from it. Otherwise, we continue to next state i+1.  
     We note that going over state 1 +action right, we do it already for reaching state 2.  
     Thus, the number of steps needed to go over all state action pairs can be calculated as follows:  
     and we can check it’s correct for all n using induction (base: n=2 => N=2).
  2. Since we want to maximize the rewards which are exponentially reduced as times goes by due to discount factor, the optimal policy is to select the path that gives us as much as possible consecutive rewards in the near future.   
     Meaning, we will go from stage 1 to n and back to 1 in order to collect these consecutive rewards (that are reduced constantly by discount factor) and then collect rewards as we did on the previous section (1->2->1->2->3->1…).  
     If we look at the summary of steps, we get the same amount as in previous section (just in a different order of execution in order to collect the maximum reward)
  3. As we saw in section d. in case n=3 the rewards vector and transition matrix (under the stochastic policy is as follows):

Applying the closed solution of the 3 equations:  
 we get:

* 1. The state space:  
     }}  
     Therefore, we have   
     The state’s defines the remaining jobs that were not served yet.  
     The start state is   
       
     The action space:  
     }  
     The action index defines the job that the server tries to serve at that time index.  
     It cannot be an index that doesn’t belong to the state at that time index.  
     Transition probabilities:  
     As stated, We note that we cannot have an action that is not part of the current state.  
     If we define some state as . We can move from to by selecting . if the job was not served (with probability ) we stay in the same state.

For all .And all other probabilities are 0 (we can’t move to other states if we took action k).

The total cost can be computed as follows:

Meaning, only if we have in current time index an unhandled job we pay with its cost.  
We can see that the state cost doesn’t depend on action we take next, but only on the current state .  
From some time, there are no jobs to handle and the cost is 0.

We can also think of it with a termination state that is the empty set (no more jobs to process).  
  
The bellman equation is therefore:

We can write it as:

When is

The optimal policy according to Bellman equation is:  
We need to check the suggested optimal policy:  
by policy evaluation.  
If we develop the policy evaluation equation according to this selected action. we get:  
We also remember that is . Thus:  
And also we can see that:

as optimal for we get:

and if we continue this way until the termination state we get:

We can sort the indexes chosen by the given policy and denote

Which match the selection according to:  
If we assume k jobs not handled in start stage, we can reorder the summation order of the cost as follows:

And from (\*\*\*) we get:  
Next, we put things together in the bellman equation we started from and apply the first action selection according to policy(selected action is :

If we put our solution to the V according to the given policy we get a valid equation:

We conclude that the given policy solves the bellman equation and thus optimal.

**3.**

**4.**

**5.**