Eric Boris

Report for Part A

Using the menu commands, set up the MDP for TOH with 3 disks, no noise, one goal, and living reward=0. The agent will use discount factor 1. From the Value Iteration menu select "Show state values (V) from VI", and then select "Reset state values (V) and Q values for VI to 0".

Use the menu command "1 step of VI" as many times as needed to answer these questions:

1a. How many iterations of VI are required to turn 1/3 of the states green? (i.e., get their expected utility values to 100).

> 4 iterations.

1b. How many iterations of VI are required to get all the states, including the start state, to 100?

> 8 iterations.

1c. From the Value Iteration menu, select "Show Policy from VI". (The policy at each state is indicated by the outgoing red arrowhead. If the suggested action is illegal, there could still be a legal state transition due to noise, but the action could also result in no change of state.) Describe this policy. Is it a good policy? Explain. Repeat the above setup except for 20% noise.

> At each iteration, the states with utility values most recently updated to 100 have policies that point towards the states that, in the previous iteration, were updated to have utility values of 100. When all the states are explored, policy arrows often don't point in the right direction. It's not a good policy.

2a. How many iterations are required for the start state to receive a nonzero value?

> 8 iterations.

2b. At this point, view the policy from VI as before. Is it a good policy? Explain.

> With 20% noise, the policy is much better and already directly leads to the goal state.

2c. Run additional VI steps to find out how many iterations are required for VI to converge. How many is it?

> VI converges after 56 iterations.

2d. After convergence, examine the computed best policy once again. Has it changed? If so, how? If not, why not? Explain. Now try simulating the agent following the computed policy.

Using the "VI Agent" menu, select "Reset state to s0". Then select "Perform 10 actions". The software should show the motion of the agent taking the actions shown in the policy. Since the current setup has 20% noise, you may see the agent deviate from the implied plan. Run this simulation 10 times, observing the agent closely.

The best policy hasn't changed because the policy converged before the values.

3a. In how many of these simulation runs did the agent ever go off the plan?

Counting any and all deviations from the plan on any given run as a 1.

runs 1 through 10 = [0, 1, 1, 1, 0, 0, 1, 1, 0, 0] sum of runs 1 through 10 = 5

3b. In how many of these simulation runs did the agent arrive in the goal state (at the end of the golden path)?

runs 1 through 10 = [1, 0, 1, 1, 1, 1, 0, 0, 1, 1] sum of runs 1 through 10 = 7

3c. For each run in which the agent did not make it to the goal in 10 steps, how many steps away from the goal was it?

runs 1 through 10 = [0, 2, 0, 0, 0, 0, 1, 2, 0, 0] sum of runs 1 through 10 = 5

3d. Are there parts of the state space that seemed never to be visited by the agent? If so, where (roughly)?

Yes. There are two of the three corners of the triangle shaped state space that are less well explored than the corner nearest the goal state. The bottom left corner near the start state is less well explored and the upper center corner where a stack of disks is made on the center peg is also less well explored.

Report for Part B

Set up the MDP with 2 disks, with 20% noise, and one goal R=100. Find a set of parameters (you can choose any combination of living reward, discount, QL-params), so that after you train for 1000 iterations, the policy shows the golden path.

4a. Report the set of parameters you chose.

Initialize with the following to establish the optimal path:
File -> Restart with 2 disks
Q-Learning -> Reset state to s0
Q-Learning -> Show Policy from QL
MDP Noise -> 20%

MDP Rewards -> One Goal, R=100
MDP Rewards -> Living R= -0.01
MDP Rewards -> Show golden path (optimal solution)
Discount -> γ = 0.9
Q-Learning -> Reset state values (V) and Q values for QL to 0
QL Params -> Fixed α=custom α
QL Params -> Fixed ε=0.2
Q-Learning -> Train for 1000 transitions

4b. If you choose an alternative value for each parameter one at a time and train for 1000 iterations each time, how does the optimal policy change? In other words, how sensitive is the trained policy to each parameter (i.e., living reward, discount, α and ε)?

Note that between each step
Q-Learning -> Reset state values (V) and Q values for QL to 0
Q-Learning -> Reset state to s0
are being selected.

Change MDP Rewards -> Living R= +0.1
The policy of the last node before the goal no longer points to the goal.

Change MDP Rewards -> Living R=0
The policy of the last node before the goal no longer points to the goal.

Change MDP Rewards -> Living R= -0.1
Optimal path remains the same.

Change MDP Rewards -> Living R= -0.01
Change Discount -> γ = 0.99
Optimal path remains the same.

Change Discount -> γ = 0.9
Optimal path remains the same.

Change Discount -> γ = 1.0
Optimal path remains the same.

Change Discount -> γ = 0.5
Optimal path remains the same.

Change Discount -> γ = 0.99
Change QL Params -> Fixed α=0.1
The policies of the second and last node along the golden path no longer point along the golden path.

Change QL Params -> Fixed α=0.2
The policies of the second and last node along the golden path no longer point along the golden path.

Change QL Params -> Fixed α= custom α
Change QL Params -> Fixed ε=0.1
The policy of the start node, the first node on the golden path, doesn't point along the golden path.

Change QL Params -> Fixed ε= custom ε
The policy of the second node on the golden path no longer points along the golden path.

4c. Now try to set the MDP with 3 disks and still with 20% noise. Try to train for 1000 iterations under a few different settings of parameters. How far are the resulting policies from the golden path? What extra steps would you try in order to improve the training results? Overall reflections.

Initialize with the following to establish the optimal path:
File -> Restart with 3 disks
Q-Learning -> Reset state to s0
Q-Learning -> Show Policy from QL
MDP Noise -> 20%
MDP Rewards -> One Goal, R=100
MDP Rewards -> Living R= -0.01
MDP Rewards -> Show golden path (optimal solution)
Discount -> γ = 1.0
Q-Learning -> Reset state values (V) and Q values for QL to 0
QL Params -> Fixed α=0.2
QL Params -> Fixed ε=0.2
Q-Learning -> Train for 1000 transitions

Note that between each step
Q-Learning -> Reset state values (V) and Q values for QL to 0
Q-Learning -> Reset state to s0
are being selected.

Change MDP Rewards -> Living R= +0.1
Almost none of the policy arrows point in the right direction.

Change MDP Rewards -> Living R=0
Almost none of the policy arrows point in the right direction.

Change MDP Rewards -> Living R= -0.1
Optimal path remains the same.

Change MDP Rewards -> Living R= -0.01
Change Discount -> γ = 0.99
Optimal path remains the same.

Change Discount -> γ = 0.9
Optimal path remains the same.

Change Discount -> γ = 0.5
Optimal path remains the same.

Change Discount -> γ = 0.99
Change QL Params -> Fixed α=0.1
The policies of the second points along the golden path.

Change QL Params -> Fixed α= custom α
Almost none of the policy arrows point in the right direction.

Change QL Params -> Fixed α= 0.2
Change QL Params -> Fixed ε=0.1
Almost none of the policy arrows point in the right direction.

Change QL Params -> Fixed ε= custom ε
Some of the policy arrows point in the right direction.

5a. In Value Iteration, since it is having a good policy that is most important to the agent, is it essential that the values of the states have converged?

No. Because often a policy is found well before the values converge.

5b. Comparing Value Iteration to Q learning, with the same number of updates (an update to Q(s, a) both in Value Iteration and Q learning), which one obtains a better policy? Which algorithm is more powerful? What makes the algorithm more powerful than the other?

Note an update to V(s) in Value Iteration can be written as updates for Q(s, a) for each action a. Then it takes a max over the actions to get V(s). In other words, in Value Iteration, an update to V(s), for a single state s, is equivalent with taking N updates to Q(s, a), where N is the number of valid actions.

Q-Learning seems to be the more powerful of the two in that it learns the policy as it's learning the values. Plus, Q-Learning also requires fewer given inputs to work.

5c. In Q learning, the agent has to learn the values of states by exploring the space, rather than computing with the Value Iteration algorithm. If getting accurate values requires re-visiting states a lot, how important would it be that all states be visited a lot? What parameter would be the best to control the number of re-visitations?

> It's not always important that every state be visited many times since there are often undesirable states. We can change the value of epsilon to find better and worse balances between exploration and exploitation.