LAB 5

Kumar Ayush - 2015eeb1060

Course Instructor - **Dr. Narayanan C. Krishnan**

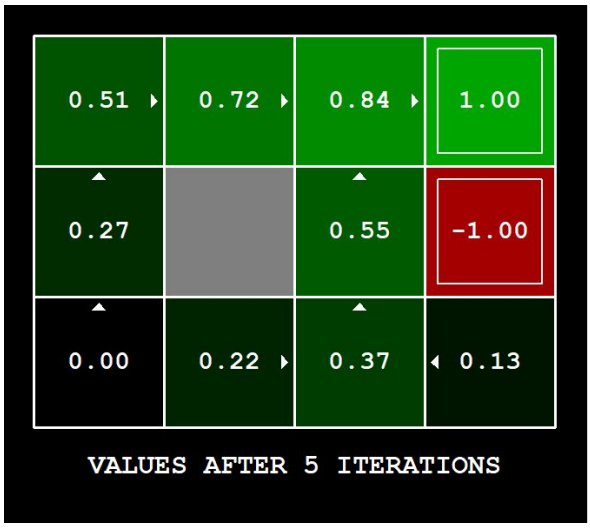
Indian Institute of Technology, Ropar

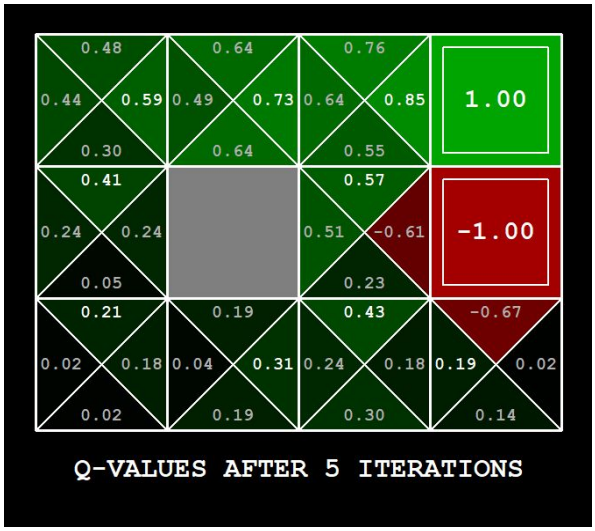
1. Value Iteration

Method

V* is computed in the *__init__* function itself using values obtained from function *computeQValueFromValues.* Rest of the function are written generic code. When no action is possible, the policy just returns *None.*

Assessments and Measures





Observation

One of the apparent observation that we see by changing the number of iterations is that the optimal action becomes constant after **10 iterations**. The policy values V* also somewhat stabilizes after **7 iterations**. The states farther from the terminal states takes longer to stabilize due to the nature of MDP algorithm.

2. Bridge Crossing Analysis

Observation

Changing noise is the effective way to cross bridge successfully, since, *we* know that the undesirable terminal state is on the sidelines. So, to avoid them, agent should

not do too much exploration but rather exploit what it already knows, that is, going always *east*.
Lowering the discount factor gives the agent less motivation to reach the desirable terminal state, so, it's left at a relatively higher value.

3. Policies

Observation

Discount factor mainly changes the exit that agent takes. Higher value promotes distant exit.
Noise promotes the amount of random exploration that agent takes.
Living reward high means agent is less likely to risk the cliff and is more likely to choose a safe path.
All three parameters works hand in hand.
3a) Discount factor limits the agent to close exit. Low living reward along with relatively low noise makes agent explore much more and risk cliffs.
3b) Discount factor is the same but the noise is relatively high so as to only take optimal path(like the previous bridge problem).
3c) Discount factor is high to make agent prefer the distant exit. Low living reward makes the agent take riskier moves.
3d) Discount factor is essentially same but noise and living reward is changed to make agent take much less number of risks.
3e) Discount factor is negligible and living reward is high making agent choosing the actions so that it never reaches any terminal states.

4. Q-Learning

Method

Computed in a similar way as in part 1.

Observation

When doing automatic learning the agent always take current optimal actions and don't explore unknown states until there's a tie. So, the training happens normally as the algorithm dictates.

5. Epsilon Greedy

Observation

Now agent takes unknown steps much more willingly but only by some probability. Setting epsilon too high(>0.5) may make agent more exploratory but it wouldn't exploit the information that it got through the said exploration. Epsilon value between (0.1 to 0.4) seems to be optimal, with of course, a trade off between exploration and exploitation.

6. Q-Learning and Pacman

Observation

There are four parameters to manipulate: number of training batches, epsilon, gamma and alpha. During testing epsilon and alpha are turned off.
Increasing number of training batches increases the win rate and vice versa.
Epsilon increases the chance that the agent may find a better path somewhere in the future and will not get stuck in local maxima. But it's a tradeoff between exploration and exploitation.
Gamma is the discount factor which works as expected. Nearer rewards are given priority.
Alpha makes recent samples more important since they encompass already learned values. Increasing it too high value will cause the agent to not give *too* much weight to newfound sample and decreasing it will cause it to not utilize at all what it has already learnt.