Reinforcement learning (CS6700)
Written assignment #2

Aravind S
EE14B013

17th Feb. 2017

1 Question 1

Here we have a gridworld of size $10x10$. Since the agent can be in any square, we have a total of 100 states. There is a start state from which the agent starts moving and there is a terminal state (goal state) which marks the end of the agent's journey. From each square, four actions can be taken - left, right, up and down. All are deterministic i.e there is no stochasticity in the environment regarding actions. So when the agent decides to move left, he will move to one square on his left (if it is possible).

A bandit problem is a simpler version of the full RL problem where the goal is to find the best arm or the optimal arm which gives the highest expected reward when pulled. Since there are totally 100 states, we can have 100 bandits - one for each state. Each bandit has 4 arms corresponding to the 4 possible actions - $\{left, right, up, down\}$ which can be taken. For squares along the edge of the world, only 3 actions can be taken - this means that the Q-value of the other action will remain 0. There is a non-zero reward of 1 if the agent reaches the goal state, otherwise there is no reward given. Also, there is a discount factor γ for the returns.

Assume for every state s, there is a bandit for estimating $Q_s(a)$ where $a \epsilon \{left, right, up, down\}$. Each $Q_s(a)$ is initialised to 0. From the start state, lets follow a sufficiently exploratory policy - equiprobable random policy or $\epsilon - greedy$ policy and generate trajectories. Update rule is as follows:

$$Q_s(a) \leftarrow Q_s(a) + \alpha(r + \gamma \frac{max}{a'} Q_{s'}(a') - Q_s(a))$$

Here s is the current state and we reach the next state s' after taking an action a in the trajectory. After sampling sufficient number of trajectories and following the above update rule, the estimates of Q converge to the optimal Q values. Note that this is very similar to Q-learning. Instead of using an action-value function $Q(s, a)$, we are using a number of $Q_s(a)$ - each of them is a bandit for all states s.

Since, Q-learning is off-policy, it doesn't matter what behaviour policy we are following. We have to make sure, sufficient exploration is done and $\alpha's$ obey stochastic averaging rules to ensure convergence and correctness of the estimate.

2 Question 2

Here, we have a 5-arm bandit problem where we know the expected payoffs of all the arms beforehand. But we don't know the mapping between each arm to its expected payoff. Expected payoffs are - 4.6, 3.1, 2.3, 1.2, 0.9. Since, we know the expected payoffs i.e $q*(a)$ for every arm we can design a small variant of UCB for this case.

$\Delta_i = q*(a*) - q*(a_i)$ where a_i denotes the arm pulled at the i^{th} time step. Now, this term can be bounded using the maximum of all possible $\Delta_i's$. Here, it can be replaced with $4.6 - 0.9 = 3.7$. This can result in better bounds than UCB.

Another variant of UCB is UCB-improved. It considers Δ_i also in the maximisation part i.e we choose the arm j which maximises $Q_j + \sqrt{\frac{2logt}{n_j}}\Delta_j$. Again bounding Δ_j in a similar fashion as mentioned above we can achieve better bounds than UCB.

In UCB, the term $u_j = \sqrt{\frac{2logt}{n_j}}$ is a measure of uncertainity in the estimate of Q_j. If we take an action more times, the estimates converge and the uncertainity in its measurement decreases. Since, we need to optimise regret we need to find the best action in the least possible time steps as we can then choose the same action again greedily to obtain zero regrets till the end. Suppose u_j falls below $0.5(4.6 - 3.1) = 0.75$, then the action with the highest estimate of Q is the optimal action and we can be sure that its expected payoff is 4.6. This is because in the worst case, arm with 3.1 payoff can have an estimate $< 0.75 + 3.1 = 3.85$ and arm with 4.6 payoff can have an estimate $> 4.6 - 0.75 = 3.85$ and in this case the arm with the higher estimate of Q is the one with 4.6 payoff and hence is the optimal arm. Since, we have provided worst-case guarantees, by sampling each action sufficient number of times we can obtain better regret bounds. We need to make sure $u_j < 0.75$ i.e $n* = 1 + ceil(\frac{2logt}{0.75^2})$ for all actions j. We need to sample each arm $n*$ times as we don't know the mapping between arms and payoffs. After pulling each arm $n*$times, we would have accumulated some regret but after that we will know which is the optimal arm. So from $t = n*k+1$ where $k = 5$ - denotes the number of arms, we can pull the optimal arm and we will obtain *zero* regret. This is another way to minimize regret.

3 Question 3

Here, we have a slightly different bandit set up. There is some unknown reward distribution for each arm. When we pull an arm, we obtain a sample from that distribution as the reward. Now when the agent picks an arm, the environment reveals the rewards which were chosen (samples from all the distributions).

Also, the definition of regret is modified here. Let t_i denote the arm which is pulled at i^{th} time step, k denote the total number of arms and r_i denote the reward samples of all the arms - vector of k values. Regret in the i^{th} time step is:

$$\Delta_i = \frac{max}{a}(r_i[a]) - r_i[t_i]$$

where $r_i[j]$ denotes the reward sample of the j^{th} arm in the i^{th} time step.

The existing algorithms works well in this setting because when we ignore the extra information which the environment gives us about other actions, it reduces to the original bandit problem with a modified regret.

Since we know the reward samples of all arms in each time step, we can utilise that information and do something better. From UCB, we need to pull the arm which maximises $Q_j + \sqrt{\frac{2logt}{n_j}}$ but since we obtain information about all arms, we can assume that all $n_j's$ are same. This means that we need to pull the arm which maximises $Q_j + constant$ i.e act greedily w.r.t current estimate. In that case, after N time-steps (N is large so that the policy converges), when the policy i.e $\frac{argmax}{j}Q_j$ converges the greedy policy is the optimal policy. Once the optimal policy is found, we pull the arm which gives the maximum expected reward and hence in an expected sense, we accumulate *zero* regret. Expected sense means - even though we determine the optimal arm the sample drawn from that arm's distribution may not be highly rewarding in every time step. But over a number of time steps, the expected regret is zero i.e Δ_i will not be *zero* but $E[\Delta_i] = 0$ after N time-steps.

4 Question 4

In this experiment, the initial estimate of Q for each arm is set to some value higher than the reward obtained when the arm is pulled. Say Q^i denote the Q for the i^{th} arm and Q^i_j denote the j^{th} estimate for Q^i. And say there are totally n arms. Here,

$$Q^i_1 = 5 \; for \; i = 1, 2, .., n$$

A greedy policy is followed here so as to learn which is the best arm to pull and hence maximise the reward obtained. Since the reward obtained after pulling an arm is just 0 or 1, the initial estimate of Q is highly optimistic i.e it is expecting a high reward when an arm is pulled.

Say initially we chose arm 1, and we obtain a reward R^1_2 which can be 0 or 1, now by stochastic averaging rule, $Q^1_2 = Q^1_2 + \alpha(R^1_2 - Q^1_2)$. Since the term $(R^1_2 - Q^1_2)$ is negative, $Q^1_2 < Q^1_1$ i.e estimate Q^1 reduces. As we are following a greedy policy other $Q^i \; for \; i \neq 1$will be higher than Q^1 and hence other arms will be pulled in the successive plays. This means there is an inherent tendency for exploration and each arm will be pulled many times (lot of exploration in the early stages) before the estimates converge. Due to this exploration inspite of following a greedy policy, there are lot of oscillations in the early stages. Once sufficient exploration is done, the value estimates start converging and $\%Optimal \, action$ increases.

So, due to this optimistic value initialisation, this method performs worse (lot of exploration) in the early stages when compared to a realistic initilisation ($Q^i_1 = 0 \; for \; i = 1, 2, .., n$) and an $\epsilon - greedy$ policy. This trick works well for stationary problems as we just need to find the optimal action once and then exploit it. For non-stationary problems, the distribution of rewards keep changing and hence we need to explore continuously as a non-optimal action can become an optimal one as time progresses.

5 Question 5

If the step-size parameters, α_n are not constant, then:

$$Q_{n+1} = Q_n + \alpha_n(R_n - Q_n) = \alpha_n R_n + (1 - \alpha_n)Q_n$$

$$Q_{n+1} = \alpha_n R_n + (1 - \alpha_n)(\alpha_{n-1} R_{n-1} + (1 - \alpha_{n-1})Q_{n-1})$$

$$Q_{n+1} = \alpha_n R_n + (1 - \alpha_n)\alpha_{n-1} R_{n-1} + (1 - \alpha_n)(1 - \alpha_{n-1})(\alpha_{n-2} R_{n-2} + (1 - \alpha_{n-2})Q_{n-2})$$

$$Q_{n+1} = \sum_{j=1}^{n} \{ \prod_{i=j+1}^{n} (1 - \alpha_i) \} \alpha_j R_j + \prod_{i=1}^{n} (1 - \alpha_i) Q_1$$

From the above equation, it is clear that in the estimate Q_n the j^{th} reward is given a weight: $\alpha_j \{ \prod_{i=j+1}^{n} (1 - \alpha_i) \}$.