Lecture 21 (Active RL)

1 Properties of Q-Learning

1. Q-Learning converges to optimal policy even if agent acts sub-optimally
2. Exploration is enough
3. Off-policy algorithm: convergence to optimal policy even if agent acts sub-optimally

2 SARSA Learning

1. Updates using (s, a, r, s', a')
2. $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha [r + \gamma Q(s', a')]$
3. More realistic when adversaries control policy
4. On-policy algorithm

3 Active Reinforcement Learning

Exploration vs exploitation trade-off

3.1 Exploration Strategy

3.1.1 ϵ Greedy

1. With small ϵ probability explore
2. Otherwise exploit
3. Exploration is very slow
4. Not very active of a strategy

3.1.2 Exploration Functions

1. They make you hallucinate
2. $f(u, n) = u + k/n$, n is number of times the state has been visited and u is the Q-value
3. States are more likely to be visited if they haven't been visited earlier
4. $Q(s, a) \leftarrow_\alpha r + \gamma \max_{a'}(f(Q(s', a'), N(s', a')))$ ($\leftarrow_\alpha$ is the update involving the previous value of $Q(s, a)$ and factor of α)

5. We can also induce ϵ greedy in this algorithm

3.1.3 Upper Confidence Bound

1. Similar to ϵ greedy but preference given to those actions which have potential to being optimal
2. $a_t = \underset{a \in A}{arg\,max} \left[\hat{Q}(a) + \sqrt{\frac{2\log t}{N_t(a)}}\right]$

3.2 Multi-arm Bandits

1. Single state but multiple actions
2. It is an example of exploration vs exploitation strategyi
3. Environment generates the probability distribution
4. Monte-Carlo evaluation is done - averaging the Q-value:

$$Q(a) = \mathbb{E}[R(a)] \approx \hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} r_t 1(a_t = a)$$

4 Problem of Generalization

1. Having too many state spaces can make RL very slow
2. Can instead use feature based representation
3. Features are combined using functions
4. This helps in generalizing the states

4.1 Linear Value Functions

$$V(s) = \sum_{i=1}^{n} w_i f_i(s)$$

$$Q(s,a) = \sum_{i=1}^{n} w_i f_i(s,a)$$

1. The goal of Q-learning is to now estimate these weights from experiences
2. Once the weights are learnt, the resulting Q-value will be close to the actual Q-value
3. For this to work efficiently, we need substantial states

4.1.1 Approximate Q-Learning

Updates happens as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(difference)$$

$$w_m \leftarrow w_m + \alpha \left[r + \gamma \max_a Q(s',a') - Q(s,a)\right] f_m(s,a)$$