Reinforcement Learning - Part 2

Alice Gao

Lecture 21

Readings: RN 21.2.3, 21.3.2, PM 12.3, 12.4, 12.7.

Outline

Learning Goals

By the end of the lecture, you should be able to

- Trace and implement the passive Q-learning algorithm.
- Trace and implement the active Q-learning algorithm.
- Compare and contrast ADP and Q-learning algorithms.
- Explain the difference between Q-learning and SARSA.

Bellman Equations for $Q(s,a)$

$Q(s,a)$ is the expected value of performing action a in state s.
We can define Bellman equations for both $V(s)$ and $Q(s,a)$.

Bellman equations for $V(s)$:

$$V(s) = R(s) + \gamma \sum_{s'} P(s'|s,a) V(s')$$

Bellman equations for $Q(s,a)$:

$$Q(s,a) = R(s) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')$$

Learning $V(s)$ and $Q(s,a)$ are equivalent!
What is the advantage of learning $Q(s,a)$?

Temporal Difference Error

Assume that we observed $\langle s_1, r_1, a, s_2 \rangle$.
Based on this transition, what should $Q(s_1, a)$ satisfy?

Start with the Bellman equations for $Q(s_1, a)$.

$$Q(s_1, a) = R(s_1) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q(s', a')$$

$Q(s_1, a)$ should be computed by the RHS of the above equation.
Assume that this transition always occurs $(P(s_2|s_1, a) = 1)$.
Thus, $Q(s_1, a)$ should be

$$R(s_1) + \gamma \max_{a'} Q(s_2, a')$$

Temporal difference (TD) error:

$$(R(s_1) + \gamma \max_{a'} Q(s_2, a')) - Q(s_1, a)$$

Q-Learning Updates

Given an experience $\langle s, r, a, s' \rangle$, update $Q(s, a)$ as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(R(s) + \gamma \max_{a'} Q(s', a') - Q(s,a)\right)$$

An alternative version:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(R(s) + \gamma \max_{a'} Q(s', a')\right)$$

Passive Q-Learning Algorithm

1. Repeat steps 2 to 4.
2. Follow policy π and generate an experience $\langle s, r, a, s' \rangle$.
3. Update reward function: $R(s) \leftarrow r$
4. Update $Q(s, a)$ by using the temporal difference update rules:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(R(s) + \gamma \max_{a'} Q(s', a') - Q(s,a) \right)$$

The learning rate α:
α controls the size of each update. If α decreases as $N(s,a)$ increases, Q values will converge to the optimal values.
For example, $\alpha(N(s,a)) = \frac{10}{9+N(s,a)}$.

Active Q-Learning Algorithm

1. Initialize $R(s), Q(s,a), N(s,a), N(s,a,s')$.
2. Repeat steps 3 to 6 until we have visited each (s,a) at least N_e times and the $Q(s,a)$ values converged.
3. Determine the best action a for current state s using $V^+(s)$.

$$a = \arg\max_a f\left(Q(s,a), N(s,a)\right),\ f(u,n) = \begin{cases} R^+,\ \text{if } n < N_e \\ u, \text{otherwise} \end{cases}$$

4. Take action a and generate an experience $\langle s, r, a, s' \rangle$
5. Update reward function: $R(s) \leftarrow r$
6. Update $Q(s,a)$ using the temporal difference update rules.

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

Properties of Q-Learning

1. Learns $Q(s,a)$ instead of $V(s)$.
2. Model-free: no need to learn the transition probs $P(s'|s,a)$.
3. Learns an approximation of the optimal Q-values as long as the agent explores sufficiently.
4. The smaller α is, the closer it will converge to the optimal Q-values, but the slower it will converge.

ADP v.s. Q-Learning

1. Requires learning the transition probabilities?

2. How much computation is performed per experience?

3. How fast does it learn?

SARSA Updates

SARSA update rule: Given an experience $\langle s, a, s', r', a' \rangle$, update $Q(s, a)$ as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \Big(R(s) + \gamma Q(s', a') - Q(s, a) \Big)$$

where a' is the actual action taken in state s'.

Q-learning update rule: Given an experience $\langle s, a, s', r' \rangle$, update $Q(s, a)$ as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \Big(R(s) + \gamma \max_{a'} Q(s', a') - Q(s, a) \Big)$$

where a' is the optimal action in state s' given current Q values.

Q-Learning v.s. SARSA

- Q-learning is off-policy whereas SARSA is on-policy.
- For a greedy agent, they are the same.
  If the agent explores, they are significantly different.
- Q-learning is more flexible: It learns to behave well even when the exploration policy is random or adversarial.
- SARSA is more realistic: It can avoid exploration with large penalties. It learns what will actually happen instead of what the agent would like to happen.
- Q-learning is more appropriate for offline learning when the agent does not explore. SARSA is more appropriate when the agent explores.

Revisiting the Learning Goals

By the end of the lecture, you should be able to

- Trace and implement the passive Q-learning algorithm.
- Trace and implement the active Q-learning algorithm.
- Compare and contrast ADP and Q-learning algorithms.
- Explain the difference between Q-learning and SARSA.