Artificial Intelligence

Lecture 08 – Reinforcement Learning

Dr. Rajiv Misra, Professor
Dept. of Computer Science & Engg.
Indian Institute of Technology, Patna
rajivm@iitp.ac.in

Contents

- Dynamic Adaptive Programming (ADP)
- Temporal Difference (TD) Learning
- Q-Learning
- SARSA

Passive Learning Agent

- Fix a policy π.
- Goal is to learn $V^{\pi}(s)$ (the expected value of policy π for state s).
- Similar to policy evaluation.
- Does not know the transition model $P(s'|s,a)$ nor the reward function $R(s)$.
- Solution: Adaptive Dynamic Programming
  - Learn $P(s'|s,a)$ and $R(s)$ using the observed transitions and rewards.
  - Learn $V^{\pi}(s)$ by solving Bellman equations (exactly or iteratively).

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

- A model-based approach

Passive ADP Algorithm

1. Repeat steps 2 to 5.
2. Follow policy π and generate an experience $\langle s, a, s', r' \rangle$.
3. Update reward function: $R(s') \leftarrow r'$
4. Update the transition probability.

$$
\begin{aligned}
N(s, a) &= N(s, a) + 1 \\
N(s, a, s') &= N(s, a, s') + 1 \\
P(s'|s, a) &= N(s, a, s')/N(s, a)
\end{aligned}
$$

5. Derive $V^{\pi}(s)$ by using the Bellman equations.

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

Passive ADP Example

s_{11}	$+1$
s_{21}	-1

- $\pi(s_{11}) = down, \pi(s_{21}) = right$
- $\gamma = 0.9$
- $R(s_{11}) = -0.04, R(s_{21}) = -0.04, R(s_{12}) = 1, R(s_{22}) = -1$
- $N(s, a) = 5, \forall s, a.$
- $N(s, a, s') = 3$ for the intended direction.
- $N(s, a, s') = 1$ for a direction to the left or right of the intended direction.
- The current state is s_{11}.

Passive ADP Example continued

experience

$<s_{11}, down, s_{21}, -0.04>$

s_{11}	$+1$
s_{21}	-1

1. No need to update the reward function.
2. Update the counts.
   $N(s_{11}, down) = 6$ and $N(s_{11}, down, s_{21}) = 4$.
3. Solve the Bellman equations.

$$V(s_{11}) = -0.04 + 0.9(0.667V(s_{21}) + 0.167(1) + 0.167V(s_{11}))$$
$$V(s_{21}) = -0.04 + 0.9(0.6(-1) + 0.2V(s_{11}) + 0.2V(s_{21}))$$

The solutions are:

$$V(s_{11}) = -0.4378, V(s_{21}) = -0.8034$$

Active ADP

The passive ADP agent learns the expected value of a fixed policy.

What action should the agent take at each step?
Two things are useful for the agent to do:

1. exploit: take an action that maximizes $V(s)$.
2. explore: take an action that is different from the optimal one.

The greedy agent seldom converges to the optimal policy and sometimes converges to horrible policies because the learned model is not the same as the true environment.

There is a trade-off between exploitation and exploration.

Trade off Exploitation and Exploration

1. ϵ-greedy exploration strategy:
   - select random action with probability ϵ, and
   - select the best action with probability $1 - \epsilon$.
   - may decrease ϵ over time.

2. Softmax selection using Gibbs/Boltzmann distribution.
   - Choose action a with probability $\frac{Q(s,a)/T}{\sum_a Q(s,a)/T}$.
   - $T > 0$ is the temperature. When T is high, the distribution is close to uniform. When T is low, the higher-valued actions have higher probabilities.

3. Initialize the values optimistically to encourage exploration.

Optimistic Utility Values to Encourage Exploration

We will learn $V^+(s)$ (the optimistic estimates of $V(s)$).

$$V^+(s) \leftarrow R(s) + \gamma \max_a f\left(\sum_{s'} P(s'|s,a)V^+(s'), N(s,a)\right)$$

$$f(u,n) = \begin{cases} R^+, \text{ if } n < N_e \\ u, \text{otherwise} \end{cases}$$

$f(u,n)$ trade-offs exploitation and exploration.

- R^+ is the optimistic estimate of the best possible reward obtainable in any state.
- If we haven't visited (s,a) at least N_e times, assume its expected value is R^+.
- Otherwise, use the current $V^+(s)$ value.

Active ADP Algorithm

1. Initialize $R(s), V^+(s), N(s,a), N(s,a,s')$.
2. Repeat steps 3 to 7 until we have visited each (s,a) at least N_e times and the $V^+(s)$ values converged.
3. Determine the best action a for current state s using $V^+(s)$.

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

4. Take action a and generate an experience $\langle s, a, s', r' \rangle$
5. Update reward function: $R(s') \leftarrow r'$
6. Update the transition probability.

$$N(s,a) = N(s,a) + 1, N(s,a,s') = N(s,a,s') + 1$$
$$P(s'|s,a) = N(s,a,s')/N(s,a)$$

7. Updated $V^+(s)$ using the Bellman updates.

$$V^+(s) \leftarrow R(s) + \gamma \max_a f\left(\sum_{s'} P(s'|s,a)V^+(s'), N(s,a)\right)$$

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 $V^+(s)$.

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

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.

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

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

- Q-Learning

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

- SARSA

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

Conclusion

Key Differences: Q-Learning vs. SARSA

Feature	Q-Learning	SARSA
Update Rule	Uses **max Q(s', a')** (off-policy)	Uses **Q(s', a') of next taken action** (on-policy)
Learning Type	More **aggressive** (optimistic updates)	More **conservative** (follows current policy)
Convergence	Faster but unstable in stochastic envs	Slower but more stable

- **Q-Learning is more optimistic** because it selects the **max Q-value** for future updates.
- **SARSA updates based on the next chosen action**, leading to more **conservative learning**.

Thank You

Dr. Rajiv Misra, Professor
Dept. of Computer Science & Engg.
Indian Institute of Technology, Patna
rajivm@iitp.ac.in