Temporal Difference & Q Learning

Sanjiban Choudhury
What if the transitions are unknown?

\[ \langle S, A, C, \mathcal{T} \rangle \]

\( s, a \)

\( s', \mathcal{T} \)
Exploration vs Exploitation

From Dan Klein
Doors

\[ a^1 \]

\[ a^2 \]

\[ a^3 \]

\[ \ldots \]

\[ ? \]

\[ ? \]

\[ ? \]
Doors

Round 1  Round 2  Round 3

\(a^1\)

\(a^2\)

\(a^3\)

-100

-1

1000
Doors

Round 1  Round 2  Round 3

\[ a^1 \]

\[ a^2 \]

\[ a^3 \]
How do we explore/exploit when picking doors?
What if we played the game over multiple time steps?
How do we estimate values of each door?
Two Ingredients of RL

Exploration Exploitation

Estimate Values $Q(s, a)$
Recap: The Swamp MDP

\[ < S, A, C, \mathcal{T} > \]

- Two absorbing states: Goal and Swamp
- Cost of each state is 1 till you reach the goal
- Let’s set \( T = 30 \)
When the MDP is known!

Run Value / Policy Iteration
When MDP is known: Policy Iteration

\[ V^\pi(s) = c(s, \pi(s)) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^\pi(s') \]\n
Estimate value

\[ \pi^+(s) = \arg \min_a c(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^\pi(s') \]\n
Improve policy
What happens when the MDP is *unknown*?
Need to **estimate the value** of policy

Value $V^\pi(s)$

Policy $\pi$
Estimate the value of policy from sample rollouts.
Estimate the value of policy from sample rollouts
Activity!
Think-Pair-Share

Think (30 sec): Given a bunch of roll-outs, how can you estimate value of a state? (Hint: More than one way!)

Pair: Find a partner

Share (45 sec): Partners exchange ideas
Option 1: Just execute the damn policy!

and look at the returns ..
Monte Carlo Evaluation

Goal: Learn $V^\pi(s)$ from complete rollout $s_1, a_1, c_1, s_2, a_2, c_2, \ldots \sim \pi$

Define: Return is the total discounted cost

$$G_t = c_{t+1} + \gamma c_{t+2} + \gamma^2 c_{t+3} + \ldots$$

Value function is the expected return

$$V^\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s]$$
Monte Carlo

Law of large numbers: \( V(s) \to V^\pi(s) \) as \( N(s) \to \infty \)

For episode in rollouts:

Increment counter \( N(s) \leftarrow N(s) + 1 \)

Increment total return
\( S(s) \leftarrow S(s) + G_t \)

Update \( V(s) = S(s)/N(s) \)
Monte Carlo

For episode in rollouts:

- Increment counter \(N(s) \leftarrow N(s) + 1\)
- Increment total return \(S(s) \leftarrow S(s) + G_t\)
- Update \(V(s) = \frac{S(s)}{N(s)}\)

Law of large numbers: \(V(s) \rightarrow V^\pi(s)\) as \(N(s) \rightarrow \infty\)
Monte Carlo

Law of large numbers: $V(s) \rightarrow V^\pi(s)$ as $N(s) \rightarrow \infty$

For episode in rollouts:

Increment counter $N(s) \leftarrow N(s) + 1$

Increment total return $S(s) \leftarrow S(s) + G_t$

Update $V(s) = S(s)/N(s)$
Exponential Moving Average MC

For episode in rollouts:

Update $V(s) \leftarrow V(s) + \alpha(G_t - V(s))$

Law of large numbers: $V(s) \to V^\pi(s)$ as $N(s) \to \infty$
Can we do better than Monte Carlo?

What if we want quick updates? (No patience to wait till end)

What if we don’t have complete episodes?
Option 2: Trust your value estimate
Temporal Difference (TD) learning

Goal: Learn $V^\pi(s)$ from traces

$(s_t, a_t, c_t, s_{t+1})$  $(s_t, a_t, c_t, s_{t+1})$  $(s_t, a_t, c_t, s_{t+1})$  $(s_t, a_t, c_t, s_{t+1})$

Recall value function $V^\pi(s)$ satisfies

$$V^\pi(s) = c(s, \pi(s)) + \gamma \mathbb{E}_{s'} V^\pi(s')$$

TD Idea: Update value using estimate of next state value

$$V(s_t) \leftarrow V(s_t) + \alpha \left( c_t + \gamma V(s_{t+1}) - V(s_t) \right)$$

Temporal Difference Error
TD Learning

For every \((s_t, a_t, c_t, s_{t+1})\)

\[ V(s_t) \leftarrow V(s_t) + \alpha (c_t + \gamma V(s_{t+1}) - V(s_t)) \]
Did you spot the trick?

\[ V^\pi(s) = c(s, \pi(s)) + \gamma \mathbb{E}_{s'} V^\pi(s') \]

\[ V(s_t) \leftarrow V(s_t) + \alpha (c_t + \gamma V(s_{t+1}) - V(s_t)) \]
Monte-Carlo

\[ V(s) \leftarrow V(s) + \alpha(G_t - V(s)) \]

- Zero Bias
- High Variance
- Always convergence
  (Just have to wait till heat death of the universe)

Temporal Difference

\[ V(s) \leftarrow V(s) + \alpha(c + \gamma V(s') - V(s)) \]

- Can have bias
- Low Variance
- May not converge if using function approximation
We have been talking about trying to learn the value of a given policy $\pi$

$$V^\pi(s) / Q^\pi(s, a)$$

What if we wanted to learn the optimal value function

$$V^*(s) / Q^*(s, a)$$
Q-learning: Learning off-policy

For every \((s_t, a_t, c_t, s_{t+1})\)

\[
Q^*(s_t, a_t) = Q^*(s_t, a_t) + \alpha(c(s_t, a_t) + \gamma \min_{a'} Q^*(s_{t+1}, a') - Q^*(s_t, a_t))
\]
Is this ... magic?

We just learned in IL how distribution shift is a big deal ...

It’s not magic. Q-learning relies on a set of assumptions:

1. Each state-action is visited infinite times
2. Learning rate $\alpha$ must be annealed over time
QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation
Q-learning: Learning off-policy

For every \((s_t, a_t, c_t, s_{t+1})\)

\[
Q^*(s_t, a_t) = Q^*(s_t, a_t) + \alpha (c(s_t, a_t) + \gamma \min_{a'} Q^*(s_{t+1}, a') - Q^*(s_t, a_t))
\]
Large-scale Q-learning with continuous actions (QT-Opt)

stored data from all past experiments \( \{(s_i, a_i, s'_i)\}_i \)

live data collection

Bellman updaters
compute \( Q_T(s, a) = r + \max_{a'} Q_\theta(s', a') \)

training buffers
- off-policy \((s, a, s', r)\)
- on-policy \((s, a, s', r)\)
- labeled \((s, a, Q_T(s, a))\)

training threads
\[
\min_\theta \| Q_\theta(s, a) - Q_T(s, a) \|^2
\]
Making Q-learning better!

Problem: Q-learning suffers from an estimation bias $\min_{a'} Q^*(s_{t+1}, a')$

Solution: Double Q-learning $Q^*(s_{t+1}, \arg\min_{a'} \tilde{Q}(s_{t+1}, a'))$

Problem: Q-learning samples uniformly from replay buffer

Solution: Prioritized DQN - samples states with higher bellman error

Problem: Q-learning doesn’t seem to learn ....

Solution: Start with high exploration + learning rate, anneal!
tl;dr

Estimate the value of policy from sample rollouts

Monte-Carlo

\[ V(s) \leftarrow V(s) + \alpha(G_t - V(s)) \]

Temporal Difference

\[ V(s) \leftarrow V(s) + \alpha(c + \gamma V(s') - V(s)) \]

Zero Bias

Can have bias

High Variance

Low Variance

Always convergence

May not converge if using function approximation

Q-learning: Learning off-policy

For every \((s_t, a_t, c_t, s_{t+1})\)

\[ Q^*(s_t, a_t) = Q^*(s_t, a_t) + \alpha(c(s_t, a_t) + \gamma \min_{a'} Q^*(s_{t+1}, a') - Q^*(s_t, a_t)) \]

Notice we are not approximating \(Q^*(s_t, a_t)\)

We don’t even care about \(\pi\)

We can learn from any data!