Temporal Difference & Q Learning

Sanjiban Choudhury
The story thus far ...
What if the transitions are unknown?

$\langle S, A, C, \mathcal{T} \rangle$

$s, a$

$s', s$
Activity!
Think-Pair-Share

Think (30 sec): What is the MDP $<S, A, C, T>$ for this robot? Is the transition $T$ known?

Pair: Find a partner

Share (45 sec): Partners exchange ideas
Model-Based OR Model Free?

Model Free

Directly learn $\pi$ or $Q(s, a)$

Model Based

Learn a model $T(s'|s, a)$, plan with model to find $\pi$
Model-Based OR Model Free?

Model Free

Directly learn $\pi$ or $Q(s, a)$

Model Based

Learn a model $P(s'|s, a)$, plan with model to find $\pi$
WHAT MAKES IT HARDER THAN REINFORCEMENT LEARNING?
Exploration vs Exploitation

From Dan Klein
Doors

\[ a^1 \]

\[ a^2 \]

\[ a^3 \]

?  

?  

?  

+100

+1

-1000
Doors

Round 1

$\alpha^1$

Round 2

$\alpha^2$

Round 3

$\alpha^3$

+100

+1

-1000
Doors

$\alpha^1$

$\alpha^2$

$\alpha^3$

Round 1  Round 2  Round 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?
\[ t = 1 \quad t = 2 \]
How do we estimate values of each door?
Two Ingredients of RL

Exploration Exploitation

Estimate Values $Q(s, a)$
Two Ingredients of RL

Exploration Exploitation

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

\(< S, A, C, T >\)

- 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

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

\[ \pi^+(s) = \arg \min_a c(s, a) + \gamma \mathbb{E}_{s' \sim T(s,a)} V^\pi(s') \]

Estimate value

Improve policy
What happens when the MDP is *unknown*?
Need to estimate the value of policy $V^\pi(s)$.
Estimate the value of policy from sample rollouts

Roll outs

Policy $\pi$
Estimate the value of policy from sample rollouts

Roll outs

Value $V^\pi(s)$
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: Trust only the actual 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

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

For episode in rollouts:

If state $s$ is visited for \textit{first} time $t$

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

Law of large numbers: $V(s) \rightarrow V^\pi(s)$
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: Believe in Bellman
Recap: Value of a state

\[ V^\pi(S_t) = c_t + \gamma c_{t+1} + \gamma^2 c_{t+2} + \ldots \]

**Expected discounted sum of cost from starting at a state and following a policy from then on**
Recap: Value of a state-action

\[ Q^\pi(s_t, a_t) = c_t + \gamma c_{t+1} + \gamma^2 c_{t+2} + \cdots \]

Expected discounted sum of cost from starting at a state, executing action and following a policy from then on

\[ Q^\pi(s_t, a_t) = c(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathcal{T}(s_t, a_t)} V^\pi(s_{t+1}) \]
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
$$V^*(s) / Q^*(s, a)$$
QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation
Training time

Reward: Grasp success determined by subtracting pre and post-drop images

Distributed RL

State, Action, Reward

Learned weights

Inference time

State: 472x472 Image and gripper aperture

Critic Function

Q(State, Action)

Q-Values

Action proposals

Cross-Entropy Method

arg max Q(State, Action)
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
\[ \{(s_i, a_i, s'_i)\}_i \]

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

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

training threads
\[ \min_\theta \| Q_\theta(s, a) - Q_T(s, a) \|^2 \]

live data collection

Kalashnikov, Irpan, Pastor, Ibarz, Herzong, Jang, Quillen, Holly, Kalakrishnan, Vanhoucke, Levine. QT-Opt: Scalable Deep Reinforcement Learning of Vision-Based Robotic Manipulation Skills
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
When things fail!

Continuous Gridworld

Optimal path

True value function

What happens when we run value iteration with a quadratic?
What happens when we run value iteration with a quadratic?

Iteration 43

The following examples from [3] Boyan and Moore, 1995 demonstrate this problem. All figures from Boyan et. al.

Example: 2D gridworld

Figure 8.1: The continuous gridworld domain.

Figure 8.1.2: Training with discrete value iteration. However, figure 8.1.3 shows that Fitted Value Iteration with quadratic regression fails to converge. The quadratic function, in trying to both be flat in the middle of state space and bend down toward 0 at the goal corner, must compensate by underestimating the values at the corner opposite the goal. These underestimates then amplify on each iteration, as the one-step lookaheads indicate that points can lower their expected cost-to-go by stepping farther away from the goal.
What happens when we run value iteration with a *quadratic*?
The problem of Bootstrapping!

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

Exploration Exploitation

Estimate Values $Q(s,a)$

Monte-Carlo

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

Zero Bias

High Variance

Always convergence (Not true in most realistic settings)

Temporal Difference

$V(s) \leftarrow V(s) + \alpha (r + \gamma V(s') - V(s))$

Can have bias

Low Variance

May not converge if using function approximation

Q-learning: Learning off-policy

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

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