Reinforcement Learning

CS 59300: RL1

September 9, 2025

Joseph Campbell

Department of Computer Science

PURDUE UNIVERSITY

Today’s lecture

1. Unknown models: Temporal Difference learning

Some content inspired by David Silver’s UCL RL course and Katerina Fragkiadaki’s CMU 10-403

Unknown models: Temporal Difference learning

Recap: Monte Carlo on-policy learning

Two-step iterative algorithm. Randomly initialize policy π…

- Evaluate the policy with sampled episodes

$$Q_\pi(s, a) \text{ approximated with empirical means}$$

- Improve the policy by acting ϵ-greedily with respect to V_π

$$\pi' = \epsilon\text{−greedy } Q(s, a)$$

Note: consider decaying ϵ to converge to an optimal policy

Recap: Monte Carlo policy evaluation

To obtain action-value empirical means instead of state-value...

Whenever state-action (s, a) is visited in an episode,

1. Increment visitation counter: $N(s, a) \leftarrow N(s, a) + 1$
2. Increment total return: $S(s, a) \leftarrow S(s, a) + G_t$

Estimate value by mean return: $Q(s, a) = S(s, a)/N(s, a)$

Recap: Monte Carlo policy evaluation

To obtain action-value empirical means instead of state-value...

Whenever state-action (s, a) is visited in an episode,

1. Increment visitation counter: $N(s, a) \leftarrow N(s, a) + 1$
2. Increment total return: $S(s, a) \leftarrow S(s, a) + G_t$

Estimate value by mean return: $Q(s, a) = S(s, a)/N(s, a)$

To compute G_t we must have complete episodes!

The problem with complete episodes

Monte Carlo learning requires complete episodes to estimate Q_π

What are the problems with this?

The problem with complete episodes

Monte Carlo learning requires complete episodes to estimate Q_π

What are the problems with this?

The problem with complete episodes

1. Value estimates take a long time to make
   - It takes time to sample from an environment! What if our environment takes 1 million steps to end?
   - If we can't update our value estimate until the episode ends, that means we spend all 1 million steps with an un-updated policy (inefficient)

2. Value estimates have high variance
   - Environments with lots of randomness means our estimates may vary wildly, meaning it takes lots of samples to form an accurate estimate

Can we learn from incomplete episodes?

Can we learn from incomplete episodes?

Yes! Bootstrapping!

Instead of computing value estimates for the *actual* return, compute them for the *estimated* return

First: re-formulate empirical value estimate

Our original value estimate explicitly calculated the empirical mean

Whenever state-action (s, a) is visited in an episode,

1. Increment visitation counter: $N(s, a) \leftarrow N(s, a) + 1$
2. Increment total return: $S(s, a) \leftarrow S(s, a) + G_t$

Estimate value by mean return: $Q(s, a) = S(s, a)/N(s, a)$

First: re-formulate empirical value estimate

Instead…let's incrementally calculate the mean

Whenever state-action (s, a) is visited in an episode,

1. Increment visitation counter: $N(s, a) \leftarrow N(s, a) + 1$
2. Update value estimate: $Q(s, a) = Q(s, a) + \frac{1}{N(s,a)} \underbrace{(G_t - Q(s, a))}_{\text{Error between our previous estimate and the observed new return}}$

First: re-formulate empirical value estimate

Instead…let's incrementally calculate the mean

Whenever state-action (s, a) is visited in an episode,

1. Increment visitation counter: $N(s, a) \leftarrow N(s, a) + 1$
2. Update value estimate: $Q(s, a) = Q(s, a) + \frac{1}{N(s,a)}(G_t - Q(s, a))$

We can generalize this to: $Q(s, a) = Q(s, a) + \alpha\,(G_t - Q(s, a))$

- Useful if actual values change over time and we want to forget

First: re-formulate empirical value estimate

Instea

Essentially an exponential moving average. Smaller α places higher priority on older values. Higher α places higher priority on more recent values. Thus "forgetting" older values.

Wher

1. In

2. U $a))$

$N(s,a)$

We can generalize this to: $Q(s,a) = Q(s,a) + \alpha\,(G_t - Q(s,a))$

- Useful if actual values change over time and we want to forget

Temporal Difference policy evaluation

In Monte Carlo learning, our "target" is the actual return

$$Q(s,a) = Q(s,a) + \alpha\,(G_t - Q(s,a))$$

In Temporal Difference learning, our target is the *estimated* return

$$Q(s,a) = Q(s,a) + \alpha\,(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s,a))$$

Why? Remember the Bellman expectation equations!

Recap: Recursive form of policy returns

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots$$

$$= r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + \cdots)$$

$$= r_{t+1} + \gamma G_{t+1}$$

This relationship allows us to decompose value functions

Recap: Bellman expectation equation

We can decompose value functions into two parts:

- The immediate reward
- The expected future returns

$$\mathbb{E}[G_t | s_t = s] = \mathbb{E}[r_{t+1} + \gamma G_t | s_t = s]$$

State-value: $V_\pi(s) = \mathbb{E}[r_{t+1} + \gamma V_\pi(s_{t+1}) | s_t = s]$

Action-value: $Q_\pi(s, a) = \mathbb{E}[r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a]$

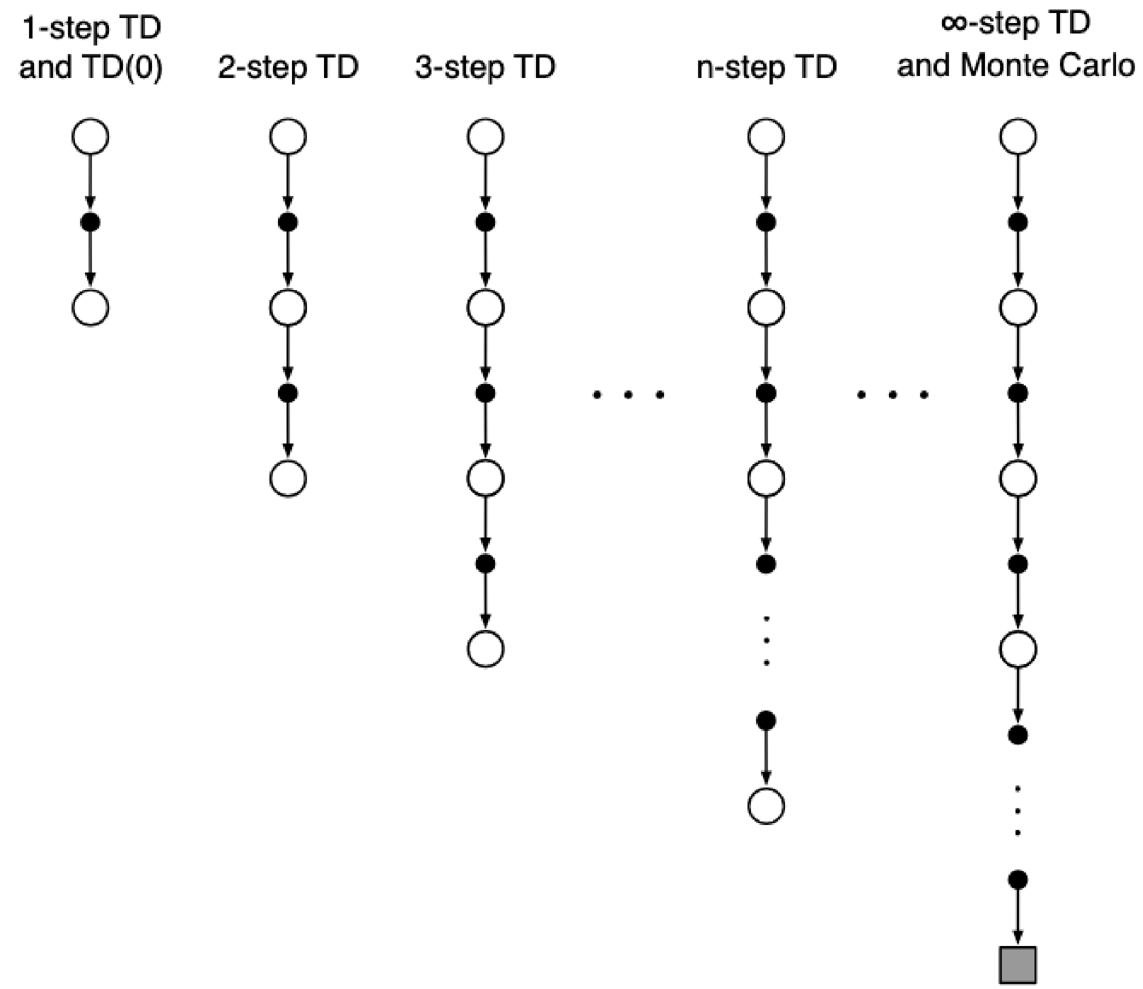
Temporal Difference policy evaluation

$$Q(s,a) = Q(s,a) + \alpha\,(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s,a))$$

$r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$ is referred to as the "TD target"

$r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s,a)$ is referred to as the "TD error"

N-step TD evaluation (if we want to)



The credit assignment problem

One of the central problems in RL is “credit assignment”

- What is it?

The credit assignment problem

One of the central problems in RL is “credit assignment”

- What is it?
- **Which actions and states in a sequence contributed to the eventual rewards?**

TD handles this by propagating reward signals backwards across multiple updates (albeit inefficiently)

N-step returns and TD(λ) can help with this (see Sutton and Barto)

The credit assignment problem

One of the central problems in RL is “credit assignment”

- Wha
- **Whi e**
  eve

How does Monte Carlo handle credit assignment?

TD ha ss
multiple updates (albeit inefficiently)

N-step returns and TD(λ) can help with this (see Sutton and Barto)

Monte Carlo vs Temporal Difference

Monte Carlo

- Can't learn until final outcome is obtained from the episode
- Can't learn *without* outcome
- Only works when episodes terminate

Temporal Difference

- Can learn after every step
- Can learn without outcome
- Can work with non-terminating episodes (lifelong learning?)

Monte Carlo vs Temporal Difference

Monte Carlo

- High *variance* in value estimate, but low *bias* (why?)
- Good convergence, but typically takes many samples
- Not sensitive to initial estimate

Temporal Difference

- Low *variance* in value estimate, but high *bias* (why?)
- Less-good convergence, but takes fewer samples
- Sensitive to initial estimate

SARSA: TD on-policy learning

Look familiar? Same as before…

Two-step iterative algorithm. Randomly initialize policy π…

- Evaluate the policy with sampled episodes

$$Q_\pi(s,a) \text{ approximated TD estimate}$$

- Improve the policy by acting ϵ-greedily with respect to Q_π

$$\pi' = \epsilon\text{−greedy } Q(s,a)$$

SARSA: TD on-policy learning

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Repeat (for each step of episode):
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ε-greedy)
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

From Sutton and Barto Chapter 6.4, which is why notation is slightly different.

SARSA: TD on-policy learning

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$

Repeat

Initia

Choo

Repe

T

C ... y)

$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$

$S \leftarrow S'$; $A \leftarrow A'$;

until S is terminal

Side note: why is it called SARSA?

From Sutton and Barto Chapter 6.4, which is why notation is slightly different.

SARSA: TD on-policy learning

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$

Repeat

Initia

Choo

Repe

Ta

C y)

$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$

$S \leftarrow S'$; $A \leftarrow A'$;

until S is terminal

Side note: why is it called SARSA?

State, action, reward, state, action,...

(nobody ever said CS people were good at naming things)

From Sutton and Barto Chapter 6.4, which is why notation is slightly different.

Temporal Difference learning is important!

“If one had to identify one area as central and novel to reinforcement learning, it would undoubtedly be temporal difference (TD) learning.”

- Sutton and Barto, Chapter 6

TD value estimates underly nearly every major critic and actor-critic method in modern reinforcement learning

- DQN, Rainbow, (MA)PPO, SAC, (MA)DDPG, TD3, Q-mix, COMA, VDN, A2C, A3C,…
- The only ones that *don’t* are pure policy search and MCTS methods
- With this, we have the foundation to discuss modern RL research!