Reinforcement Learning: Value Functions

BIOE 498/598 PJ

Spring 2021

Review

Last time

- RL agents learn by trial and error.
- RL problems are formulated as MDPs.
- Monte Carlo methods can find policies for RL problems.

Review

Last time

- RL agents learn by trial and error.
- RL problems are formulated as MDPs.
- Monte Carlo methods can find policies for RL problems.
- **Today:** How do random simulations lead to optimal policies?

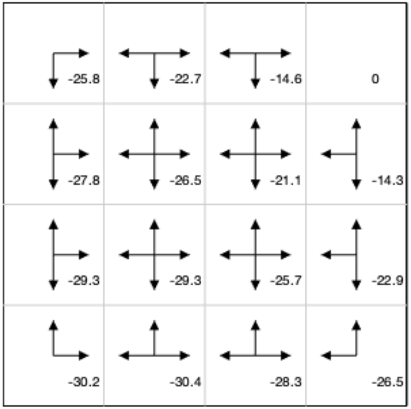
A Monte Carlo approach for Gridworld

- Each grid square is a state.
- Actions: move up, down, left, or right, but the agent cannot leave the grid.
- Reward: -1 for each step.
- Policy: Random.

Starting from a random state, make random moves until the agent reaches the end.

Repeat may times and average the total rewards from each trajectory.

The policy is to move to squares with better Monte Carlo returns.



Value functions

- We are using Monte Carlo to learn a **value function**.
- The value of a state is the expected reward from that state to the end of the trajectory:
$$V(s_i) = \mathbb{E}\left\{\sum_{k=i}^{T} r_k\right\} = \mathbb{E}\{R_i\}$$
where R_i is the *return* starting at state s_i, i.e. the cumulative reward for the rest of the trajectory: $R_i = r_i + r_{i+1} + \cdots + r_{T-1} + r_T$.

Value functions

- We are using Monte Carlo to learn a **value function**.
- The value of a state is the expected reward from that state to the end of the trajectory:
$$V(s_i) = \mathbb{E}\left\{\sum_{k=i}^{T} r_k\right\} = \mathbb{E}\{R_i\}$$
where R_i is the *return* starting at state s_i, i.e. the cumulative reward for the rest of the trajectory: $R_i = r_i + r_{i+1} + \cdots + r_{T-1} + r_T$.
- If we know the value function we can derive a policy: Take the action that moves to the state with the highest value.

Trajectories

- A trajectory in an MDP is a sequence of states, actions, and rewards:

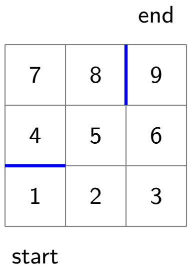
$$s_0, a_0, r_0,\ s_1, a_1, r_1,\ \ldots, s_{T-1}, a_{T-1}, r_{T-1},\ s_T, r_T$$

- The length T can vary for every trajectory.
- There is no action selected in the terminal state s_T, but there can be a terminal reward r_T.
- A reward r_i can be positive (reward), negative (penalty), or zero. Some MDPs only have a nonzero terminal reward!

From trajectories to value functions

Let's calculate $V(s)$ for a 3×3 Gridworld board.

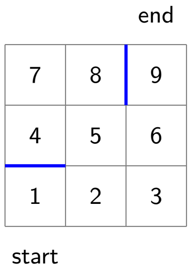
The MDP is deterministic, so knowing s_i and s_{i+1} tells us a_i. Also, $r_i = -1$ for all $0 \leq i < T$.



From trajectories to value functions

Let's calculate $V(s)$ for a 3×3 Gridworld board.

The MDP is deterministic, so knowing s_i and s_{i+1} tells us a_i. Also, $r_i = -1$ for all $0 \leq i < T$.



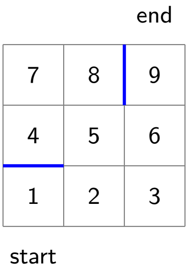
Four trajectories beginning at s_1:

$$\begin{aligned}
\tau_1 &: \quad 1,2,5,4,5,6,3,6,9 & R_{\tau_1} &= -8 \\
\tau_2 &: \quad 1,2,3,6,3,2,5,8,7,8,5,6,9 & R_{\tau_2} &= -12 \\
\tau_3 &: \quad 1,2,5,2,3,6,9 & R_{\tau_3} &= -6 \\
\tau_4 &: \quad 1,2,5,4,5,2,3,6,5,8,5,6,3,2,5,6,9 & R_{\tau_4} &= -16
\end{aligned}$$

From trajectories to value functions

Let's calculate $V(s)$ for a 3×3 Gridworld board.

The MDP is deterministic, so knowing s_i and s_{i+1} tells us a_i. Also, $r_i = -1$ for all $0 \leq i < T$.



Four trajectories beginning at s_1:

$$\begin{aligned}
\tau_1 &: \quad 1,2,5,4,5,6,3,6,9 & R_{\tau_1} &= -8 \\
\tau_2 &: \quad 1,2,3,6,3,2,5,8,7,8,5,6,9 & R_{\tau_2} &= -12 \\
\tau_3 &: \quad 1,2,5,2,3,6,9 & R_{\tau_3} &= -6 \\
\tau_4 &: \quad 1,2,5,4,5,2,3,6,5,8,5,6,3,2,5,6,9 & R_{\tau_4} &= -16
\end{aligned}$$

$$V(s_1) \approx \frac{R_{\tau_1} + R_{\tau_2} + R_{\tau_3} + R_{\tau_4}}{4} = \frac{(-8)+(-12)+(-6)+(-16)}{4} = -10.5$$

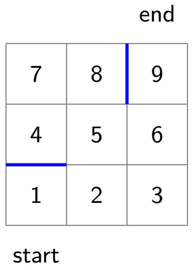
Re-using our trajectories to find $V(s_2)$

τ_1 : $1, 2, 5, 4, 5, 6, 3, 6, 9$

τ_2 : $1, 2, 3, 6, 3, 2, 5, 8, 7, 8, 5, 6, 9$

τ_3 : $1, 2, 5, 2, 3, 6, 9$

τ_4 : $1, 2, 5, 4, 5, 2, 3, 6, 5, 8, 5, 6, 3, 2, 5, 6, 9$



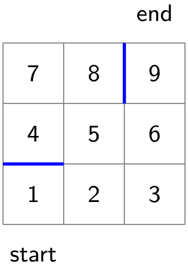
Re-using our trajectories to find $V(s_2)$

τ_1 : $1, 2, 5, 4, 5, 6, 3, 6, 9$

τ_2 : $1, 2, 3, 6, 3, 2, 5, 8, 7, 8, 5, 6, 9$

τ_3 : $1, 2, 5, 2, 3, 6, 9$

τ_4 : $1, 2, 5, 4, 5, 2, 3, 6, 5, 8, 5, 6, 3, 2, 5, 6, 9$



We can estimate $V(s_2)$ using the same trajectories because of the Markov Property. Every visit to s_2 is equivalent to new trajectory that begins at s_2.

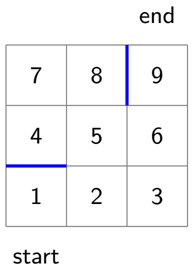
Re-using our trajectories to find $V(s_2)$

$\tau_1 :\quad 1, 2, 5, 4, 5, 6, 3, 6, 9$

$\tau_2 :\quad 1, 2, 3, 6, 3, 2, 5, 8, 7, 8, 5, 6, 9$

$\tau_3 :\quad 1, 2, 5, 2, 3, 6, 9$

$\tau_4 :\quad 1, 2, 5, 4, 5, 2, 3, 6, 5, 8, 5, 6, 3, 2, 5, 6, 9$



We can estimate $V(s_2)$ using the same trajectories because of the Markov Property. Every visit to s_2 is equivalent to new trajectory that begins at s_2.

Some trajectories visit s_2 more than once. For example, τ_3 has two returns $R = -5$ and $R = -3$.

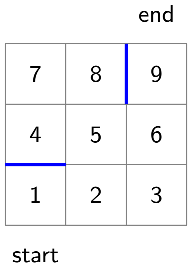
Re-using our trajectories to find $V(s_2)$

τ_1 : $1, 2, 5, 4, 5, 6, 3, 6, 9$

τ_2 : $1, 2, 3, 6, 3, 2, 5, 8, 7, 8, 5, 6, 9$

τ_3 : $1, 2, 5, 2, 3, 6, 9$

τ_4 : $1, 2, 5, 4, 5, 2, 3, 6, 5, 8, 5, 6, 3, 2, 5, 6, 9$



We can estimate $V(s_2)$ using the same trajectories because of the Markov Property. Every visit to s_2 is equivalent to new trajectory that begins at s_2.

Some trajectories visit s_2 more than once. For example, τ_3 has two returns $R = -5$ and $R = -3$.

Every-visit average: $(-7 - 11 - 7 - 5 - 3 - 15 - 11 - 3)/8 = -7.75$.
Last-visit average: $(-7 - 7 - 3 - 3)/4 = -5$.

Every-visit vs. last-visit

- Gridworld is deterministic and the agent has complete control over its actions.
- The agent should never visit the same state twice. Whatever sequence of actions are optimal after the second visit should have been selected after the first visit.
- For these problems, the last-visit estimate is closest to optimal.

Every-visit vs. last-visit

- Gridworld is deterministic and the agent has complete control over its actions.
- The agent should never visit the same state twice. Whatever sequence of actions are optimal after the second visit should have been selected after the first visit.
- For these problems, the last-visit estimate is closest to optimal.

- Stochastic problems can revisit the same state under the optimal policy.
- Imagine a stochastic airline, where planes fly to an unknown destination after takeoff.
  - If you are trying to fly to Champaign, flights departing from O'Hare have the best chance of landing in Champaign.
  - The optimal policy would be go back to O'Hare even if you've already been there.

Can we always find optimal policies from a value function?

Yes. Acting *greedy* with respect to a value function is optimal. Recall our definition of the value function

$$V(s_i) = \mathbb{E}\{r_i + r_{i+1} + \cdots + r_T\}.$$

Can we always find optimal policies from a value function?

Yes. Acting *greedy* with respect to a value function is optimal. Recall our definition of the value function

$$V(s_i) = \mathbb{E}\{r_i + r_{i+1} + \cdots + r_T\}.$$

At any state s_i, the optimal policy follows the objective

$$\begin{aligned}
&\max_{a_i} \mathbb{E}\{r_i + r_{i+1} + \cdots + r_T\} \\
=&\max_{a_i} \mathbb{E}\{r_i\} + \mathbb{E}\{r_{i+1} + \cdots + r_T\} \\
=&\max_{a_i} \mathbb{E}\{r_i\} + V(s_{i+1})
\end{aligned}$$

Can we always find optimal policies from a value function?

Yes. Acting *greedy* with respect to a value function is optimal. Recall our definition of the value function

$$V(s_i) = \mathbb{E}\{r_i + r_{i+1} + \cdots + r_T\}.$$

At any state s_i, the optimal policy follows the objective

$$\begin{aligned}
&\max_{a_i} \mathbb{E}\{r_i + r_{i+1} + \cdots + r_T\} \\
=&\max_{a_i} \mathbb{E}\{r_i\} + \mathbb{E}\{r_{i+1} + \cdots + r_T\} \\
=&\max_{a_i} \mathbb{E}\{r_i\} + V(s_{i+1})
\end{aligned}$$

For Gridworld, $r_i = -1$ for all states, so the optimal policy at state s_i satisfies

$$\max_{a_i} V(s_{i+1})$$

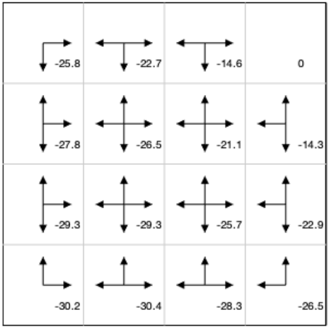
i.e., select the action that brings the agent to the state with the largest value.

Limitations of pure Monte Carlo

Random walks generate inefficient trajectories, as evidenced by the Monte Carlo value functions.

We can tell that some actions are bad even before we have a true value function.

Ideally, we would use early trajectories to speed up later trajectories.



Limitations of pure Monte Carlo

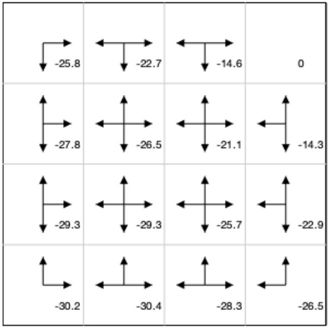
Random walks generate inefficient trajectories, as evidenced by the Monte Carlo value functions.

We can tell that some actions are bad even before we have a true value function.

Ideally, we would use early trajectories to speed up later trajectories.

One solution is **generalized policy iteration**.

1. Use a random policy π to generate trajectories and estimate $V(s)$.
2. Adjust π to be *greedy* for $V(s)$.
3. Repeat (1–2) until π stops changing.



$$\begin{tikzcd}
\pi(s,a) \arrow[r, bend left, "\text{policy evaluation}"] & V(s) \arrow[l, bend left, "\text{policy improvement}"]
\end{tikzcd}$$

Does generalized policy iteration converge to an optimal policy?

Yes, policy iteration is guaranteed to find optimal policies *provided every state is visited an infinite number of times*.

In practice, policy iteration works well with finite visits, but *tabular* methods require visiting every state. Without a visit, we have no way to tell if a policy should move to that state.

Does generalized policy iteration converge to an optimal policy?

Yes, policy iteration is guaranteed to find optimal policies *provided every state is visited an infinite number of times*.

In practice, policy iteration works well with finite visits, but *tabular* methods require visiting every state. Without a visit, we have no way to tell if a policy should move to that state.

For Gridworld, we started trajectories in every state to ensure a visit, but this was inefficient.

Does generalized policy iteration converge to an optimal policy?

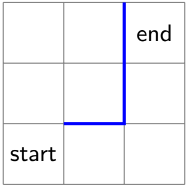
Yes, policy iteration is guaranteed to find optimal policies *provided every state is visited an infinite number of times*.

In practice, policy iteration works well with finite visits, but *tabular* methods require visiting every state. Without a visit, we have no way to tell if a policy should move to that state.

For Gridworld, we started trajectories in every state to ensure a visit, but this was inefficient.

For this board, why would we every visit the states in the upper left?

No optimal policy would ever move "up" at the starting position.



Does generalized policy iteration converge to an optimal policy?

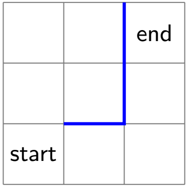
Yes, policy iteration is guaranteed to find optimal policies *provided every state is visited an infinite number of times*.

In practice, policy iteration works well with finite visits, but *tabular* methods require visiting every state. Without a visit, we have no way to tell if a policy should move to that state.

For Gridworld, we started trajectories in every state to ensure a visit, but this was inefficient.

For this board, why would we every visit the states in the upper left?

No optimal policy would ever move "up" at the starting position.



Next time we'll learn *online* methods to estimate a value function as we move through an MDP.

Summary

- **Policy evaluation** computes $V(s)$ for a given policy.
- **Policy iteration** makes greedy improvements to a policy.
- If you don't have a good starting policy, behave randomly.
- Tabular methods track $V(s)$ for every state in the MDP. They require visiting every state many times and storing the results.