Reinforcement Learning Training 2025

Model-Free Approach

Motivation

Recall in policy iteration

$$v_{k+1}(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_k(s') \right]$$

- To make this work, we need to know the model dynamics or $p(s', r|s, a)$.
- However, we do now know p.
- Instead, we will resort to *sampling*.
  - Collecting experience by following some policy in the real world or running the agent through a policy in simulation.

Model-Free Learning

- Monte Carlo (MC) methods
- Temporal difference (TD) methods

Monte Carlo

- We use the law of large numbers (LLN) from statistics.
  - Average of samples is a good estimate for the actual unknown quantity.
  - This estimate becomes better and better as the number of trials of the experiment (samples) increases.

Monte Carlo

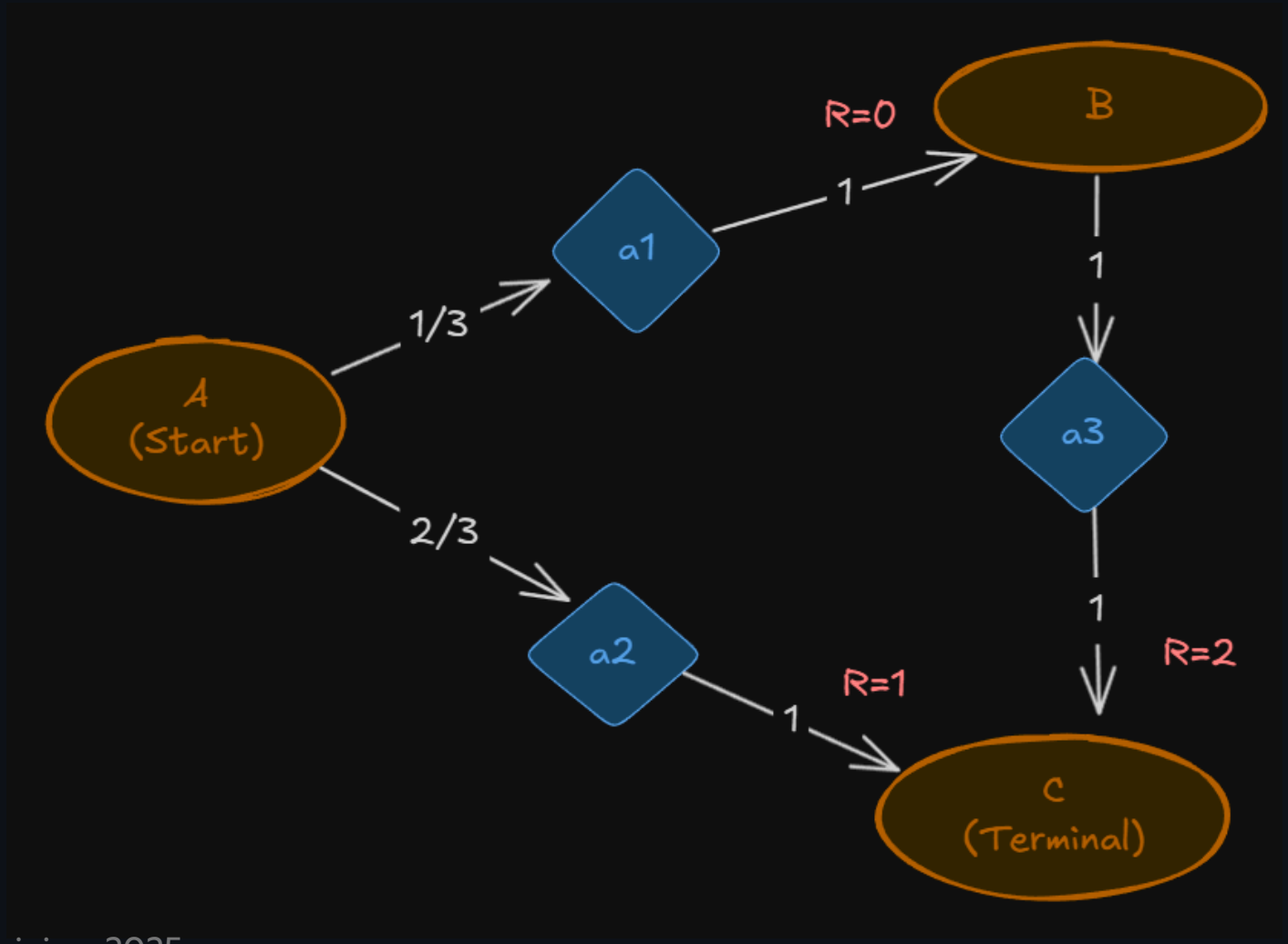
- Recall that We want to calculate

$$v_\pi(s) = \mathbf{E}_\pi[G_t | S_t = s]$$

- We let the agent start from this state $S_t = s$, follow the policy π to take actions, and keep doing so until termination.
  - We call one round of actions an **episode**.
- We record the total sum of rewards for each episode.
- We average the rewards to get an estimate of $v_\pi(s)$ for the policy π.

> MC methods replaces expected returns with the average of sample returns.

Worked Example



Solution v

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_\pi(s')\right]$$

let $\gamma = 1$

$V(C) = \boxed{0}$ (Terminal)

$V(B) = 1\left[1 \times [2 + 1(0)]\right]$

$= \boxed{2}$

$V(A) = \frac{1}{3}\left[1 \times (0 + 1(2))\right]$

$+ \frac{2}{3}\left[1 \times (1 + 1(0))\right]$

$= \frac{1}{3}(2) + \frac{2}{3}(1) = \boxed{4/3}$

Solution q

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \sum_{a'} \pi(a'|s')\, q(s',a')\right]$$

$q(a_3) = 1\,[\,2 + 1\,\Sigma(\ldots)\,]$

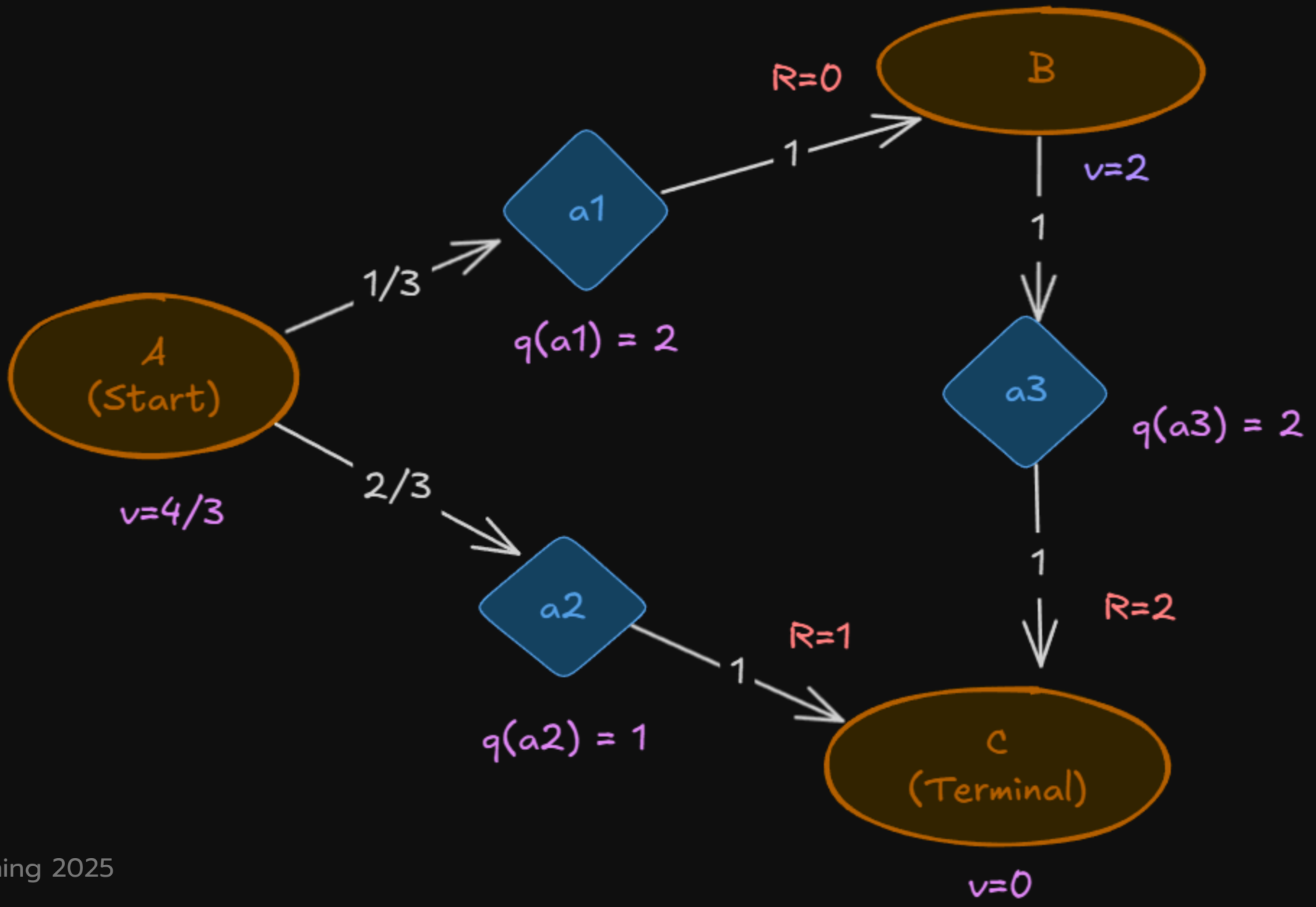
$= \boxed{2}$

$q(a_2) = 1\,[\,1 + 1\,\Sigma(\ldots)\,]$

$= \boxed{1}$

$q(a_1) = 1\,[\,0 + 1\,(\,1\,(2)\,)\,]$

$= \boxed{2}$


R=0
B
1
a1
v=2
1
1/3
A
(Start)
q(a1) = 2
a3
q(a3) = 2
2/3
v=4/3
1
R=2
a2
R=1
1
q(a2) = 1
C
(Terminal)
v=0

Estimate $v(A)$

- We simulate many episodes.

Episode	Path	Reward from A
1	A $\rightarrow$ C	G_1 = 1
2	A $\rightarrow$ B $\rightarrow$ C	G_2 = 0 + 2 = 2
3	A $\rightarrow$ B $\rightarrow$ C	G_3 = 0 + 2 = 2
4	A $\rightarrow$ C	G_4 = 1
...	...	G_n

Results

Monte Carlo estimates the value function $v(A)$ as the average return observed after visiting A.

$$v(A) = \frac{G_1 + G_2 + G_3 + G_4 + \dots}{n} = \frac{1 + 2 + 2 + 1 + \dots}{n} \to \frac{4}{3}$$

Online Method

- Instead of averaging all the returns at the end (the sample mean), we can use the incremental (update) method to estimate $v(A)$ as each new return is observed.
- This is also called the "sample-average" update and is given by:

$$v_{n+1} = v_n + \frac{1}{n}(G_n - v_n)$$

Estimate $q(a_1)$ and $q(a_2)$

Episode	Path	Actions at A	Reward from Action at A
1	A $\rightarrow$ C	a_2	$G_1 = 1$
2	A $\rightarrow$ B $\rightarrow$ C	a_1	$G_2 = 0 + 2$
3	A $\rightarrow$ B $\rightarrow$ C	a_1	$G_3 = 0 + 2$
4	A $\rightarrow$ C	a_2	$G_4 = 1$
...	...	...	G_n

Estimate $q(a_1)$ and $q(a_2)$

$$q(a_1) = \frac{G_2 + G_3 + \dots}{n} = \frac{2 + 2 + \dots}{n} \to 2$$

$$q(a_2) = \frac{G_1 + G_4 + \dots}{n} = \frac{1 + 1 + \dots}{n} \to 1$$

Online update

$$q_{n+1} = q_n + \frac{1}{n}(G_n - q_n)$$

Comparing estimations of v and q

- Notice that the calculation of v and q is the same.
- This is because both functions are fundamentally estimates of an expected value—just over different types of returns.
  - $v(s)$ - average over all times you start at s
  - $q(s, a)$ - average over all times you start at s and pick a

Monte Carlo: First-Visit vs. Every-Visit

- *First-Visit MC*
  - Only the first time a state (or state-action pair) is visited in an episode, the return following that visit is used to update the estimate.
- *Every-Visit MC*
  - Every time a state (or state-action pair) is visited in an episode, the return following that visit is used to update the estimate (even if visited multiple times within the same episode).

Control with Monte Carlo

- We will use generalized policy iteration (GPI).
  - Find q (not vs)
  - Improve policy using greedy optimization.
  - Repeat ...

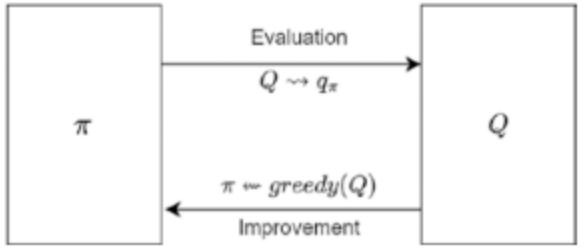


Figure 4-5. *Iteration between the two steps. The first step evaluates to get the state-action values in sync with the policy being followed. The second step performs policy improvement to do a greedy maximization for actions*

Control with Monte Carlo

- However, in MC, the greedy opimization does not work very well.
  - To be greedy, we need to know all q.
  - But being greedy, we will never *explore* all q.
- To fix this, we will use **ε-greedy policy**.

Control with Monte Carlo

- The agent exploits the knowledge with probability 1-ε and explores with probability ε.

$$\pi(a|s) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{|A|} & \text{for } a = \text{argmax}_a Q(s,a) \\ \frac{\varepsilon}{|A|} & \text{otherwise} \end{cases}$$

Control with Monte Carlo

- Next, we will not run MC until values of q converges.
- We will run MC prediction followed by policy improvement on an episode-by-episode basis.
- This way, there is no need for a large number of iterations in the estimation/prediction step

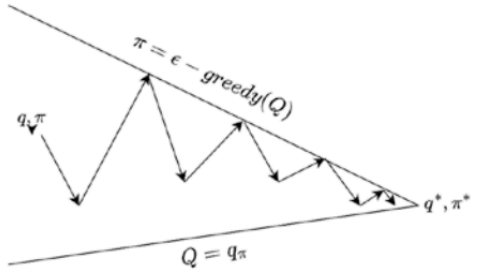


Figure 4-6. *Iteration between the two steps. The first step is the MC prediction/ evaluation for a single step to move the q-values in the direction of the current policy. The second step is that of policy improvement to do a ε-greedy maximization for actions*

Gridworld