Reinforce Learning

Monte Carlo Methods in RL

소프트웨어 꼰대 강의

노기섭 교수

(kafa46@hongik.ac.kr)

Contents

- **Recap:**
  - Dynamic Programming in RL

- **You will learn:**
  - Understand why DP is impractical in many environments
  - Explain the Monte Carlo learning principle
  - Compute "returns G_t"
  - First-Visit vs Every-Visit MC
  - MC Prediction
  - MC Control and ε-greedy exploration
  - Recognize limitations of MC methods

Recap: Dynamic Programming in RL

Limitations of Dynamic Programming in RL

1. Model Requirement: Not easy to apply

DP assumes that the **environment model is known**.

$$P(s'|s,a) \qquad R(s,a)$$

In many real-world problems,
the transition probabilities and reward functions are **unknown**.

2. State Explosion: curse of dimensionality

The number of state–action pairs grows rapidly:

$$|S| \times |A|$$

As the environment becomes more complex,
the computational cost becomes **prohibitively large**.

3. Continuous State or Action Spaces: Not real world-based

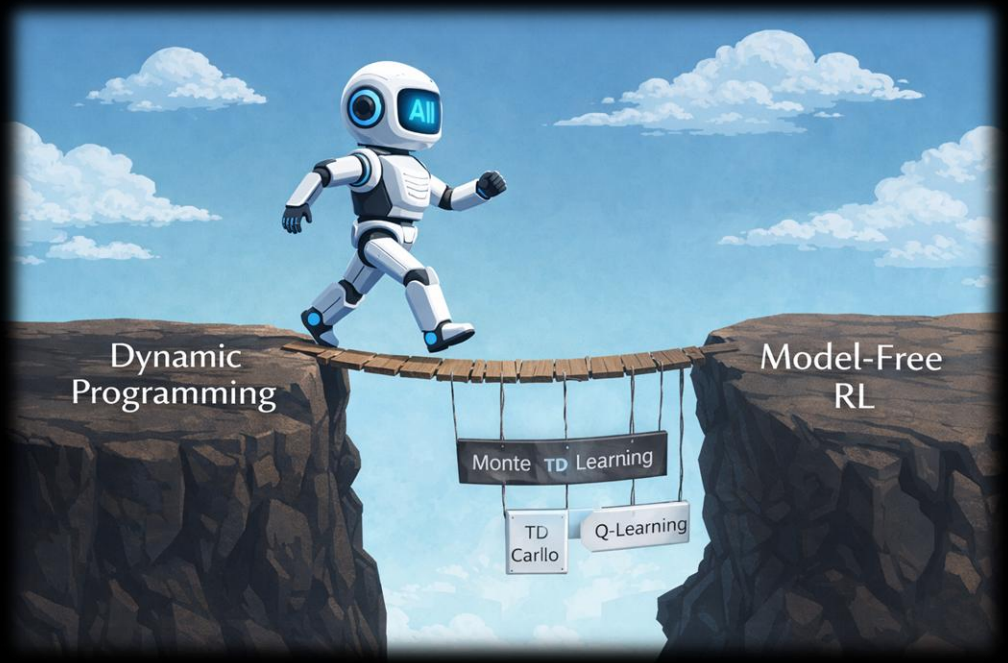
DP typically assumes **discrete states and actions**.

Bridge to Next RL

DL limitations motivate model-free reinforcement learning methods,

Such as:

- ✓ Monte Carlo (MC) methods
- ✓ Temporal Difference (TD) learning
- ✓ Q-learning



Role of DP

Dynamic Programming motivates many RL algorithms.

Dynamic Programming Concept	Reinforcement Learning Equivalent
Policy Evaluation	Temporal Difference (TD) Learning
Policy Improvement	Greedy / ε-greedy Policy
Policy Iteration	Actor–Critic Methods
Value Iteration	Q-learning

Foundations of Reinforcement Learning

Three fundamental learning paradigms:

Method	Key Characteristics
Dynamic Programming (DP)	Model-based, bootstrapping
Monte Carlo Methods (MC)	Model-free, uses full return
Temporal Difference (TD)	Model-free, bootstrapping

Evolution of Reinforcement Learning Methods

DP → MC → TD

Therefore, TD = MC + DP

TD provides efficient online learning without requiring a model.

- ✓ From MC → learning from sampled experience
- ✓ From DP → bootstrapping with value estimates

Monte Carlo

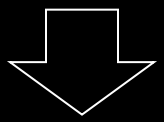
Motivation of MC

DP assumes that the **environment model is known**.

$$P(s'|s,a) \qquad R(s,a)$$

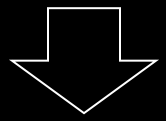
In many real-world problems,

- ✓ the transition probabilities and reward functions are **unknown**.
- ✓ The environment is only accessible through interaction.



Solution:

Learn value functions directly from experience samples.



Key idea:

Use sampled episodes instead of full environment models.

Monte Carlo Learning Idea

Learn from complete episodes!

Wait until the episode ends, then learn from the full return.

Learning rule:

Estimate value functions using sample returns.

Return definition:

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

Characteristics:

- ✓ Model-free
- ✓ No bootstrapping
- ✓ Uses **sample averages**

Property:

Learning occurs after episode terminates.

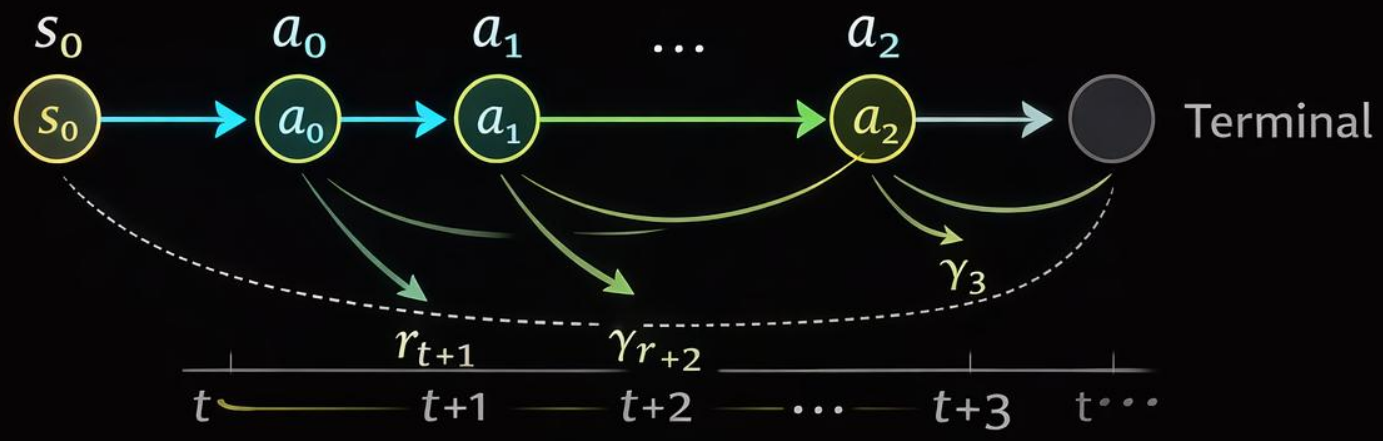
DP vs MC

Aspect	Dynamic Programming (DP)	Monte Carlo (MC)
Model	Required (known P, R)	Not required (model-free)
Learning Type	Expectation (full backup)	Sample-based (trajectory)
Update Target	Expected value over all next states	Actual return G_t
Timing	Iterative full sweeps	After complete episode
Bootstrapping	Yes	No

[참고] Bootstrapping

Image: generated by Nano Banana

Episode and Return Visualization



Trajectory

- Sequence of states, actions, rewards
- Ends in terminal state

$$G_t = r_{t+1} + \gamma r_{+2} + \gamma r_{+3} + \dots$$

r_{t+1} γ_{t+2} $\gamma^2 t_{+3}$ terminal

---- t ------- $t+2$ ----- $t+3$

Return from time t

- **Return** = total discounted **future reward** from time t
- Future rewards are discounted by γ

Return = total discounted **future reward** from **time *t***
— Future rewards are discounted by γ

Monte Carlo Prediction

Goal:

Estimate the **state value function**

$$V(s)$$

Approach:

Average the returns observed after visiting state s.

Update rule:

Update rule:

$$V(s) \leftarrow \text{Average}\,(G_t)$$

where G_t are returns observed after visiting s.

First-Visit vs. Every-Visit

Two MC Variants

Both converge to the true value function under suitable conditions!

First-Visit MC

Episode: **A** → B → C → A → D

Only the first **A** is used.

$$V(s) = \mathbb{E}[G_t | s_t = s]$$

$$= \frac{1}{N} \sum G_t$$

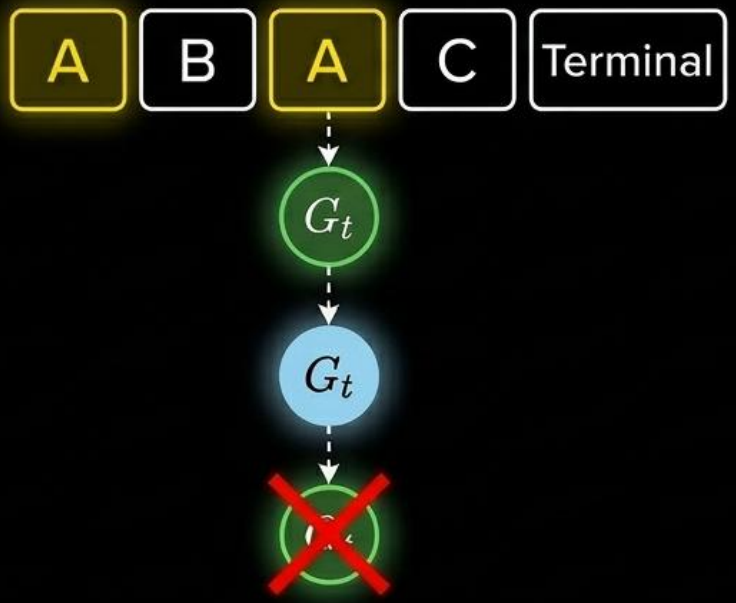
샘플 수에 따른 차이,
어차피 기댓값(평균)은
동일한 값으로 수렴!

Every-Visit MC

Episode: **A** → B → C → **A** → D

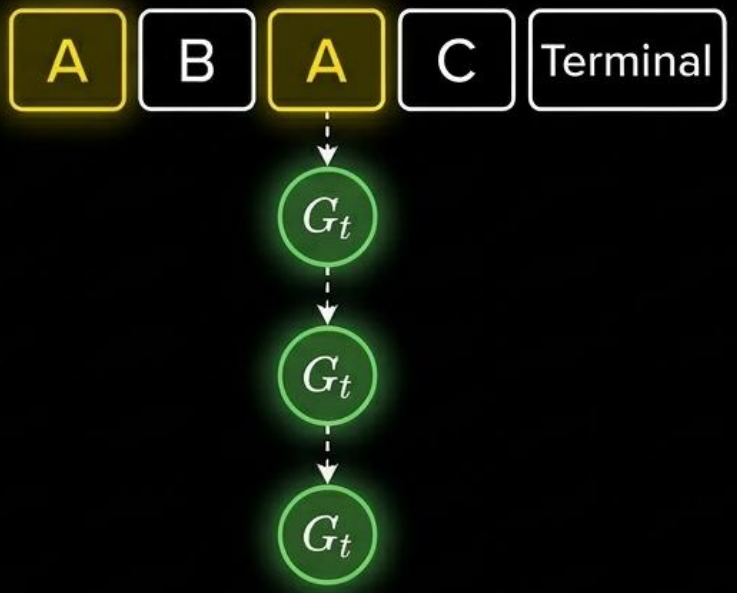
Both **A occurrences** update the value.

First-Visit vs. Every-Visit: Visual Interpretation



$V(A) \leftarrow \text{average}(G_t \text{ first})$

Uses the return following the first occurrence of a state.

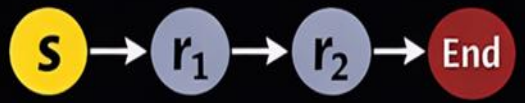
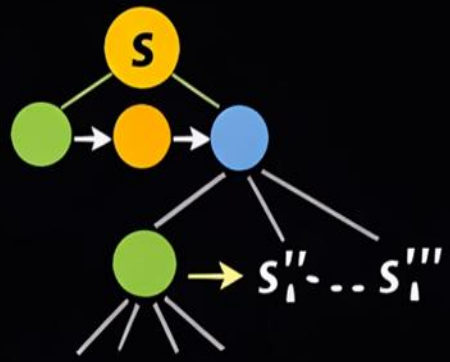


$V(A) \leftarrow \text{average}(G_t \text{ all visits})$

More details in visit types

Aspect	First-Visit MC	Every-Visit MC
Samples per episode	1 per state	N per state (N = visit count)
Variance	Relatively higher	Relatively lower (more samples)
Implementation	Requires first-visit tracking	Records all visits
Convergence speed	Potentially slower	Potentially faster (more data)
Theoretical analysis	Simpler (unbiased)	More complex
Practical usage	More commonly used	Less common

Bootstrapping in RL

Dynamic Programming (DP)	Monte Carlo (MC)	Temporal Difference (TD)
Expectation →	Sampling →	Sampling + Bootstrapping →
		
$V(s) = \mathbb{E}[R + \gamma V(s')]$	$V(s) = r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots$	$V(s) = R + \gamma V(s')$
Full Model	Sample Return	TD Update
Expectation	No Bootstrapping	Bootstrap

No Bootstrapping ⟷ Bootstrapping

MC Control: why does it need?

Goal:

Find the **optimal policy** π^*
The agent improves its policy through **interaction with the environment**.

Main Components:

Policy Evaluation

Estimate the value of current policy

$$V(s) = \mathbb{E}[G_t | s_t = s]$$

Using sample returns form episodes

Policy Improvement

Update the policy to better actions

$$\pi(s) \leftarrow \arg\max_a Q(s, a)$$

If the agent always choose the greedy action,
some actions may never be explored!

Go to next slide

MC Control: Strategy

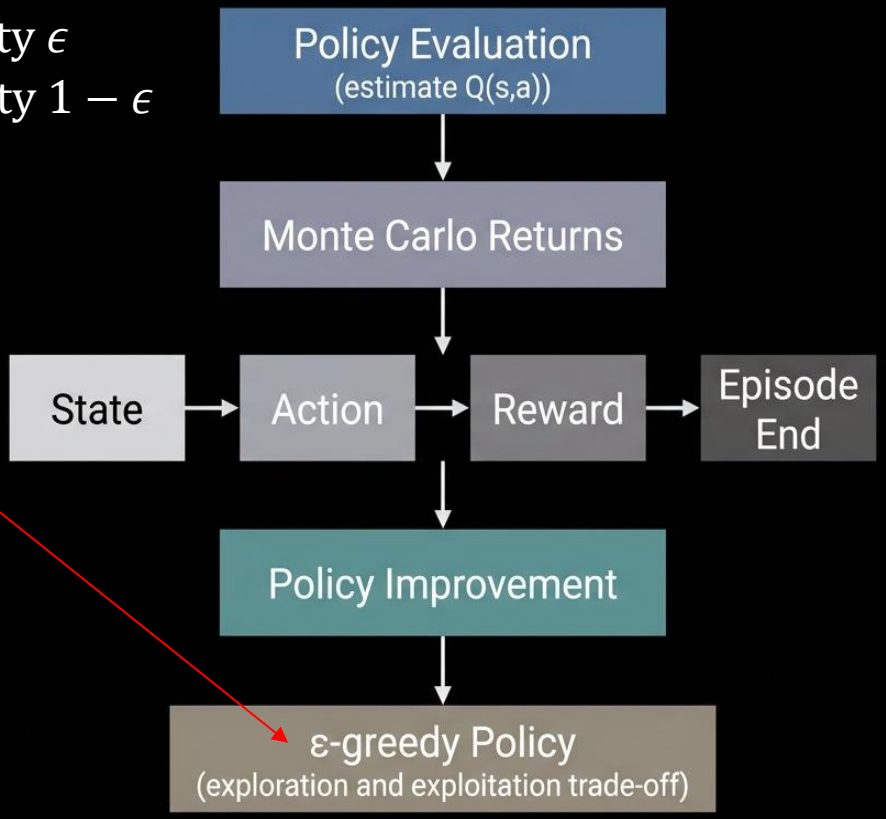
Typical Strategy: ϵ-greedy Policy

$$\pi(a|s) = \begin{cases} \text{random action} & \text{with probability } \epsilon \\ \arg\max_a Q(s,a) & \text{with probability } 1-\epsilon \end{cases}$$

Interpretation:

- $\epsilon \rightarrow$ exploration
- $1-\epsilon \rightarrow$ exploitation

This process gradually converges toward the **optimal policy**.



MC Algorithm

Initialize Q(s,a)

Repeat for each episode

1. Generate episode using ε-greedy policy
2. Compute returns G
3. Update Q(s,a)
4. Improve policy (greedy)

Advantages of MC Methods

Model-Free Learning

Does **not require an environment model**

- ✓ No transition probability $P(s'|s,a)$
- ✓ Learns directly from **experience**

Conceptually Simple

Learning is based on **sample returns**

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

- ✓ No bootstrapping
- ✓ No recursive value updates

Converges to Correct Value Estimates

Given **sufficient episodes**, MC methods converge to $V^\pi(s)$

(True expected return under policy π.)

Limitations of MC

Requires Episode Termination

MC methods must wait until the **episode ends**.

Problem:
Learning cannot occur **until the episode finishes**.

High Variance

Returns are based on **entire episode outcomes**.

Different trajectories may produce **very different returns**.

Problem:
Learning updates can be **unstable and noisy**.

Difficult for Continuing Tasks

In **continuing tasks** (no terminal state),
it becomes difficult to compute the full return.

Bridge to Temporal Difference (TD) Learning

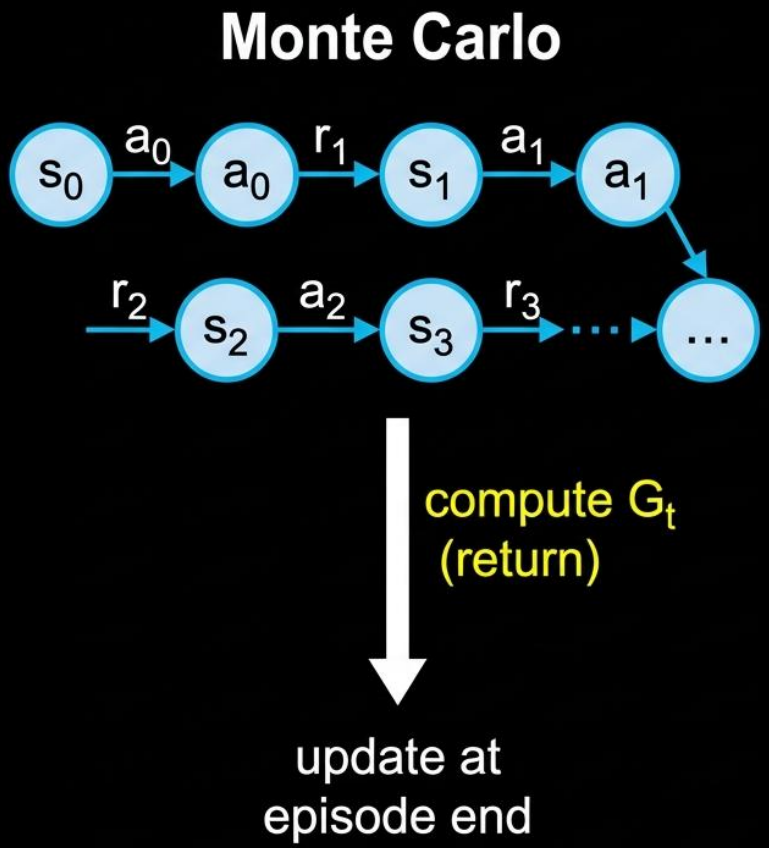
MC vs. TD

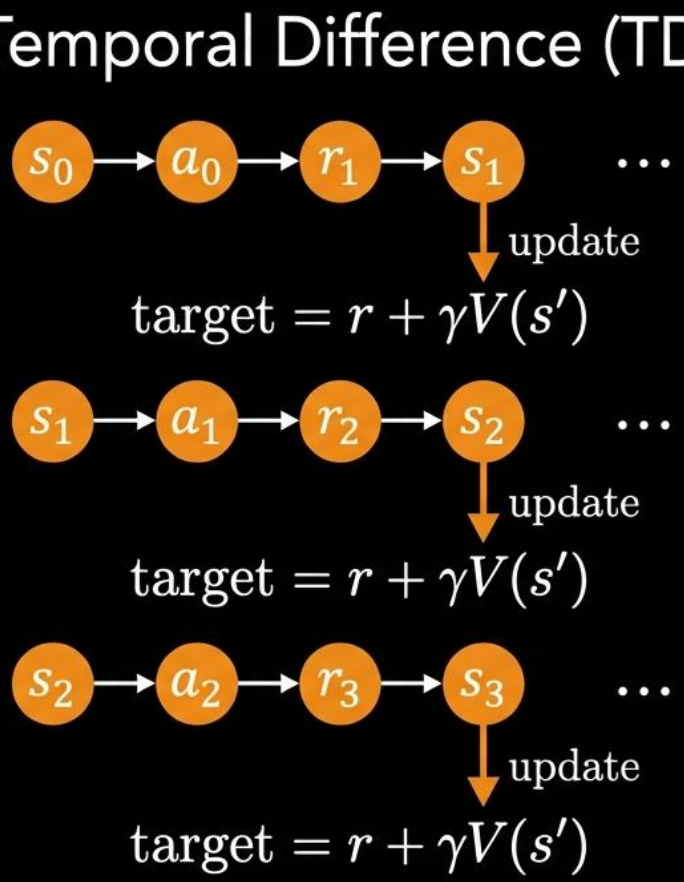
- Monte Carlo
  - Wait until episode ends
  - $V(s) \leftarrow G_t$
  - Uses actual return from the trajectory
- Temporal Difference
  - Learn step-by-step during the episode
  - $V(s) \leftarrow r + \gamma V(s)$
  - Uses actual return bootstrapped estimate of the next state value

Conceptual Understanding the Bridge (MC → TD)

MC learns from complete return.

TD learns from step-wise bootstrapping.





MC vs TD: Full Return vs Bootstrapping

MC learns from complete returns,

while TD learns by bootstrapping

Method	Learning Timing
MC	After episode ends
TD	At every time step

MC waits until the end of the episode,

whereas TD updates estimates step-by-step.

수고하셨습니다 ..^^..