Reinforce Learning

Temporal Difference in RL

소프트웨어 꼰대 강의

노기섭 교수

(kafa46@hongik.ac.kr)

Contents

- **Goal:**
  - Understand how TD learning combines ideas from MC & DP to estimate value functions efficiently.

- **You will learn:**
  - TD prediction
  - TD control
  - SARSA
  - Q-learning
  - Practical advantages of TD methods

Motivation: Why Temporal Difference?

Problem on MC

- Must wait until the end of episode
- Cannot learn from incomplete trajectories

Problem on DP

- Requires environment model (i.e., Strong Assumption!)

Temporal Difference (TD) learns:

- From incomplete trajectories
- Without a model
- Using bootstrapping

Core Idea of Temporal Difference

TD update:

$$V(s_t) \leftarrow V(s_t) + \alpha\left(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right)$$

- ✓ Prediction: $V(s_t)$
- ✓ Target: $r + \gamma(s_{t+1})$
- ✓ TD Error: $\delta = Target - Prediction$
- ✓ Update: Learning from Prediction Error

Insight:

"Bootstrapping learning"

Update current estimate using next estimate.

TD vs. MC

Approach	MC	TD
Target	G_t	$r + \gamma V(s')$
Behavior	✓ Update after full episode ✓ Low bias ✓ High variance	✓ Update every step ✓ Higher bias ✓ Lower variance

TD trades bias for lower variance

TD Error

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

Measurement: how surprising the transition was large error

Large error → value estimate was inaccurate.

Large δ → Surprise → Learning happens

TD Control 1: SARSA

SARSA (On-policy TD)

The update uses the **actual action taken by the current policy**.

The name **SARSA** comes from the transition tuple:

$$(S, A, R, S', A')$$

Update Rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$$

TD error

Learns what it actually does!

Learning Process:

1. Observe current state s
2. Choose action a using a policy (e.g., ϵ-greedy)
3. Receive reward r and next state s'
4. Choose next action a' using the policy (ϵ-greedy)
5. Update Q-value using TD error

에이전트가 실제로 행동할 때 사용하는 정책과 학습 업데이트에 사용하는 정책이 동일하다!

Uses actual next action
→ safer learning

✓ "나는 가끔 실수(탐험)를 한다"는 사실까지 고려한 정책을 학습

✓ 더 보수적인 정책을 선택: (예) 절벽에서 멀리 돌아 감 (안전 우선)

TD Control 2: Q-Learning

Q-Learning (Off-policy TD)

- ✓ Learn **optimal action-value function** independent of policy used for exploration.
- ✓ **Assumption: Agent will take the best action in the next state**

Update Rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

TD error

Learns optimal policy regardless of behavior!

Learning Process:

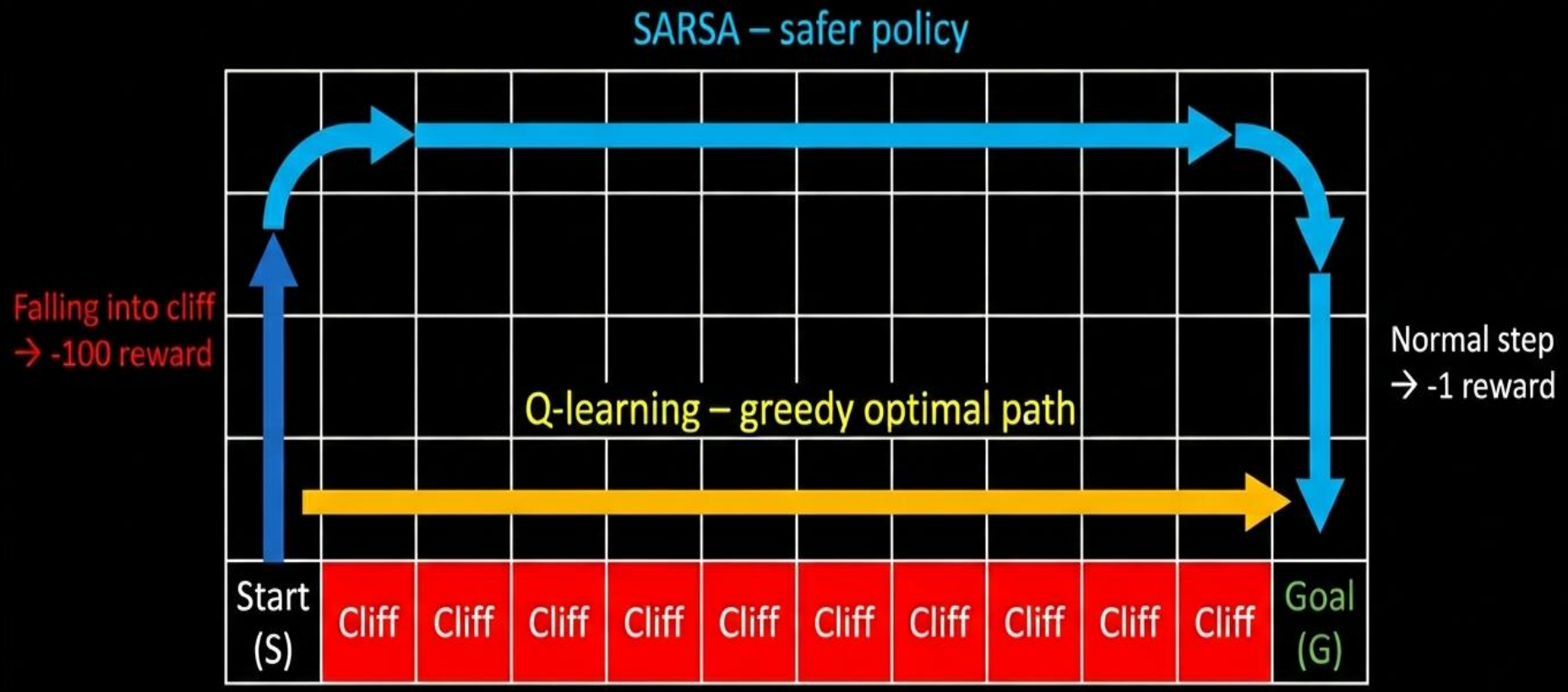
1. Observe current state s
2. Choose action a using ϵ-greedy
3. Execute action & observe r and next state s'
4. Update Q-value using the maximum possible next value (max Q)

The agent may explore randomly, but it learns the greedy optimal policy.

Uses greedy next action
→ optimal but risky

실제로는 탐험(exploration)을 위한 무작위 행동을 하더라도,
다음 상태에서는 나는 최적의 행동을 할 것이라고 가정한다.
(예) 절벽이 위험하더라도, 기댓값이 높으면 선택 (절벽 옆 최단 경로 선택!)

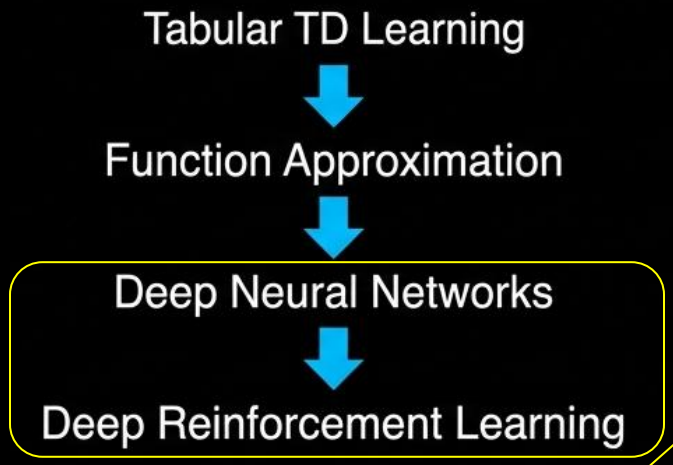
Conceptual Visualization



Bridge to Deep RL

- TD learning is the foundation of modern RL algorithms.
- Most deep RL methods use NN to approximate value functions, while still relying on TD updates.

From Tabular RL to Deep RL



Examples of Deep RL Algorithms

1) Value-based methods
DQN Double DQN Dueling DQN

2) Policy-based methods
Policy Gradient REINFORCE

3) Actor-Critic methods
A2C A3C

- ✓ High-dimensional state spaces
- ✓ Continuous control problems
- ✓ Large-scale environments

수고하셨습니다 ..^^..