Chapter Outline

Classic Single-Agent Reinforcement Learning Algorithms

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures
10.2 Consensus and Formation Control
10.3 Game Theory and RL Fundamentals
10.4 Single-Agent RL Algorithms
10.5 Multi-Agent RL and Scaling
10.6 Technologies and Challenges
10.7 RflySim Framework and Interfaces
10.8 Experiments and Validation
10.9 Exercises and Classroom Discussion

4 Units in This Section

10.4.1 Q-Learning

Textbook: Sec. 10.1.3.4, item 1
RL: reinforcement learning

10.4.2 Deep Q-Networks

Textbook: Sec. 10.1.3.4, item 2

10.4.3 Actor–Critic and DDPG

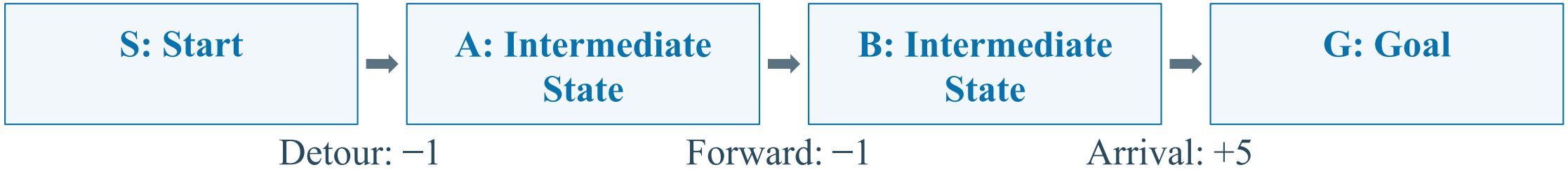
Textbook: Sec. 10.1.3.4, items 3–4
DDPG: Deep Deterministic Policy Gradient

10.4.4 Algorithm Improvements and Selection

Textbook: Sec. 10.1.3.4, item 5

10.4.1 Q-Learning: Tasks and Experience

From a Known Model to Learning from Samples



- **Teaching model:** S has direct and detour actions; A and B can only move forward. G is terminal; discount = 0.9.
- **Experience:** one observed transition, containing state, action, reward, next state and termination flag.
- **Exact reference:** at S, the direct action is worth 2 and the detour 2.15. How do estimates approach these values?

10.4.1 Q-Learning: Temporal-Difference (TD) Error; Eqs. 10.30–10.31

Update One Q-Value with a TD Error

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\delta_t \tag{10.30}$$

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \tag{10.31}$$

$$y_t = r_{t+1} + \gamma(1 - d_{t+1}) \max_{a'} Q(s_{t+1}, a')$$

$d_{t+1} = 1$ means true termination: the target is just the current reward; $0 < \alpha \leq 1$.

10.4.1 Q-Learning: A Reproducible Update

Three Numbers in One Q-Update

S: Start
Take the Detour

→ Reward: -1 →

A: Next State
Not Terminal

Experience and Estimate	Given Value
Old Q: Detour at S	1
Maximum Q at A	3.5
Discount / Learning Rate	0.9 / 0.5

Target

$$y = -1 + 0.9 \times 3.5 = 2.15$$

TD Error

$$\delta = 2.15 - 1 = 1.15$$

Updated Q-Value

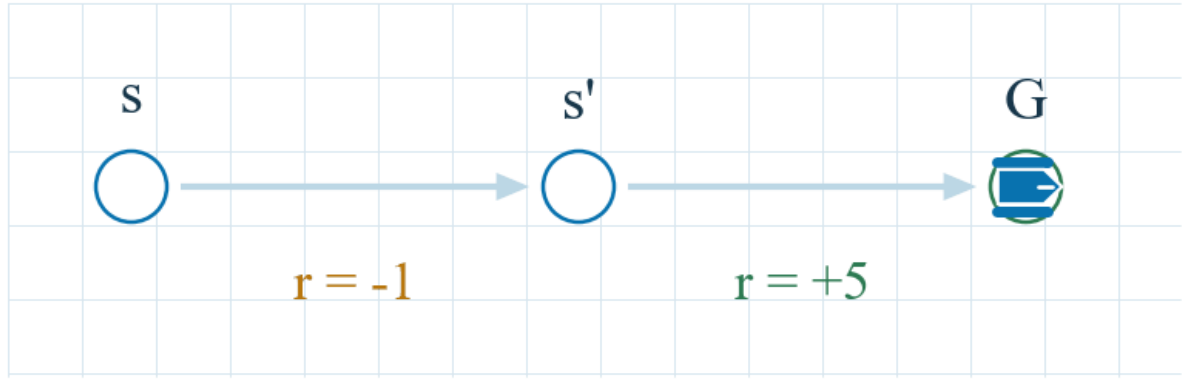
$$1 + 0.5 \times 1.15 = 1.575$$

Update only the visited entry for the detour at S; all other entries stay unchanged.

10.4.1 Q-Learning: Terminal Targets

Updates for Nonterminal and Terminal Samples

Terminal transition



s: 2 → 2.3 s': 4 → 4.5

No terminal bootstrap

y = 5 (done)

Old estimate 4

Updated 4.5



$\alpha = 0.5, \gamma = 0.9$

Nonterminal and terminal Q-value updates

One-step examples, separate from four-state navigation; no bootstrap at true termination.

10.4.1 Q-Learning: Off-Policy Learning and Exploration

Exploration Can Differ from the Learning Target

Concept	Role	Example
Behavior Policy	Generates experience	Sometimes choose a currently worse action to explore
Target Policy	Defines the policy being learned	Use the maximum next-state Q-value in the target
Off-Policy	Behavior and target policies may differ	Past transitions can be reused
On-Policy	Data come from the current or a recent sampling policy	Proximal Policy Optimization (PPO) uses the current data batch

Two actions, one uniquely greedy; $\varepsilon = 0.2$: greedy probability = $0.8 + 0.2/2 = 0.9$.

10.4.1 Q-Learning: Navigation Q-Table Updates

A Full Sampling Cycle and Its Results

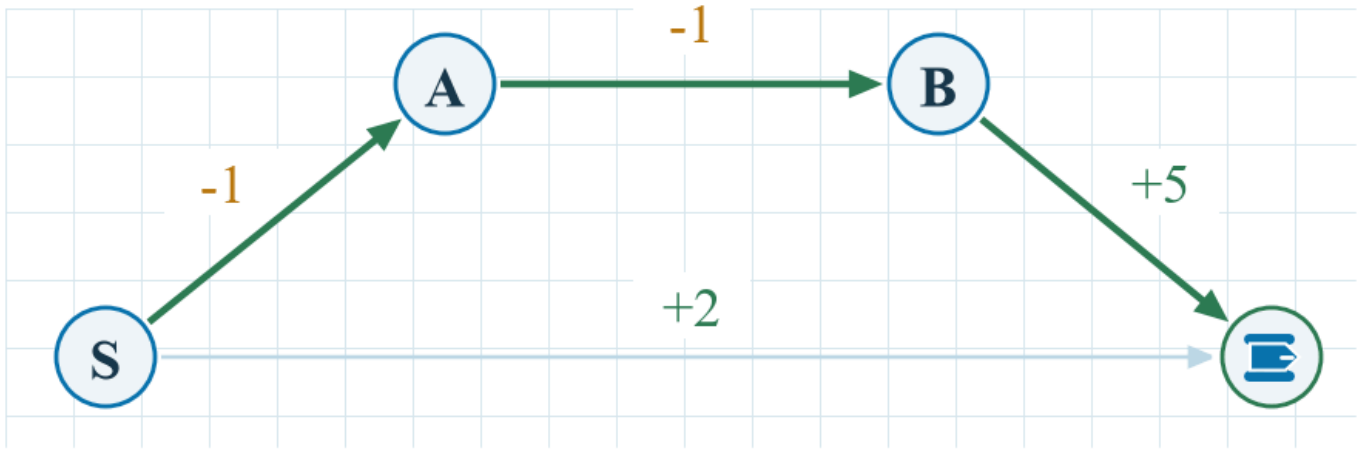
Round	S: Detour	A: Forward	B → G	S: Direct
0	0	0	0	0
1	−0.5000	−0.5000	2.5000	1.0000
4	−0.2906	2.1562	4.6875	1.8750
8	1.6002	3.3457	4.9805	1.9922
16	2.1418	3.4988	4.9999	2.0000
32	2.1500	3.5000	5.0000	2.0000

Each round: S→A, A→B, B→G, S→G.
All Q start at 0; learning rate = 0.5; discount = 0.9.

10.4.1 Q-Learning: Sampling and Path Selection

Repeated Samples Reveal the Detour's Value

Fixed sample order S→A→B→G / S→G



Both routes stay sampled $\alpha = 0.5, \gamma = 0.9$

Sample sweep 32

Q values of actions at S

Detour 2.1500

Direct 2.0000

Current choice at S:
Detour

Fixed sampling covers both paths; rounds 1, 4, 8, 16, 32 shown. All rounds are calculated.

Greedy: direct at round 8, detour at 16. Currently worse actions still need samples.

10.4.1 Q-Learning: Scope and Conditions

Convergence and Termination

- **Tabular scope:** states and actions must be enumerable. Each pair has its own estimate; high-dimensional continuous states make a table impractical.

- **Stochastic convergence:** a finite, stationary Markov decision process (MDP), bounded rewards, sufficient visits and suitably decaying learning rates.

$$\sum_k \alpha_k(s, a) = \infty, \qquad \sum_k \alpha_k^2(s, a) < \infty$$

True termination: target = current reward. Time truncation only: bootstrap from the last actual state before truncation.

Chapter Outline

Classic Single-Agent Reinforcement Learning Algorithms

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures

10.2 Consensus and Formation Control

10.3 Game Theory and RL Fundamentals

10.4 Single-Agent RL Algorithms

10.5 Multi-Agent RL and Scaling

10.6 Technologies and Challenges

10.7 RflySim Framework and Interfaces

10.8 Experiments and Validation

10.9 Exercises and Classroom Discussion

4 Units in This Section

10.4.1 Q-Learning

Textbook: Sec. 10.1.3.4, item 1

10.4.2 Deep Q-Networks

Textbook: Sec. 10.1.3.4, item 2

10.4.3 Actor–Critic and DDPG

Textbook: Sec. 10.1.3.4, items 3–4

10.4.4 Algorithm Improvements and Selection

Textbook: Sec. 10.1.3.4, item 5

10.4.2 Deep Q-Network (DQN): Fig. 10.6

How Tables and Networks Estimate Action Values

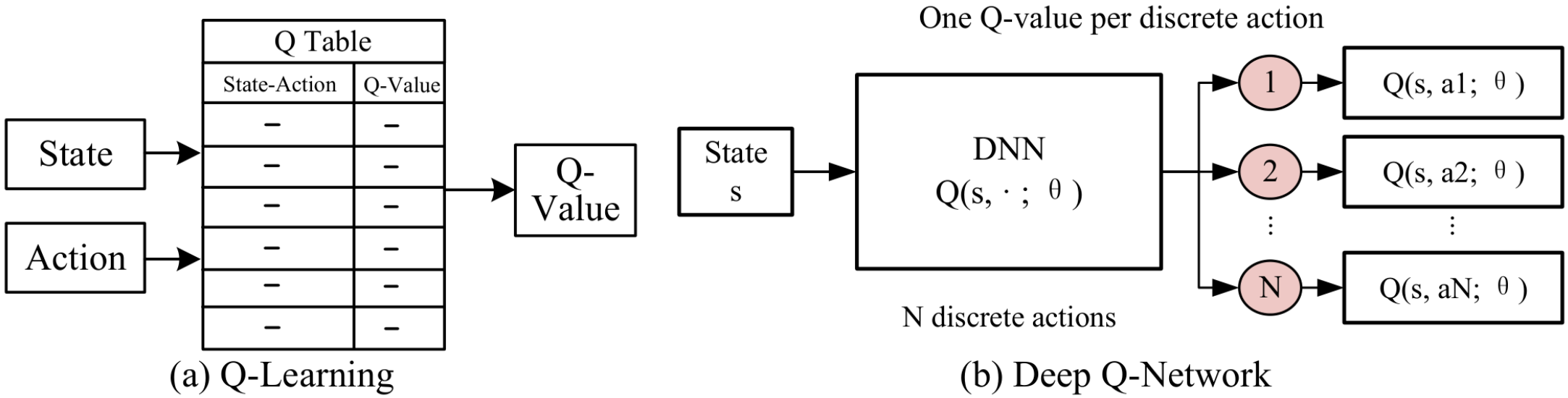


Fig. 10.6 Q-value estimation in Q-Learning and DQN

DQN approximates Q-values with shared parameters. Inputs may be high-dimensional; standard outputs represent finite discrete actions.

10.4.2 Deep Q-Networks: State, Outputs and Selection

Output Q-Values, Then Compare to Select an Action

State Input
Position / Image / Velocity

→

Online Q-Network
Three Action Values

→

Action Selection
Compare or Explore

$$Q(s,\cdot;\theta) = [1.2,\ 2.4,\ 0.8]$$

Action	Left	Forward	Right
Q-Value	1.2	2.4	0.8
Greedy Choice	No	Yes	No

10.4.2 Deep Q-Networks: Stabilizing Training

Replay and Target Networks Play Different Roles

Environment Transition
Store in Replay Buffer D → **Random Minibatch**
Reuse Past Samples → **Target Computes y**
Online Fits y

$$D\colon (s, a, r, s', d)$$

- **Replay buffer:** reduce neighboring-sample correlations and reuse experience. Unsampled actions still lack evidence.
- **Target network:** hold parameters fixed or change them slowly, so the fitted target does not immediately follow each online update.

10.4.2 Deep Q-Networks: Eqs. 10.32–10.33

DQN Targets and Squared Error

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim D}[(y - Q(s,a;\theta))^2] \tag{10.32}$$

$$y = r + \gamma \max_{a'} Q(s', a'; \theta^-) \tag{10.33}$$

$$y = r + \gamma(1-d) \max_{a'} Q(s', a'; \theta^-)$$

Eq. 10.33 is nonterminal; the last equation handles mixed batches. Hold the target fixed during this update.

10.4.2 Deep Q-Networks: Eq. 10.34 and One Update

Gradient Descent Moves the Prediction Toward Its Target

$$\theta \leftarrow \theta - \alpha \nabla_\theta (y - Q(s, a; \theta))^2 \tag{10.34}$$

$$Q = \theta = 1, \quad y = 2.15, \quad \mathcal{L} = 1.3225$$

$$\nabla_\theta \mathcal{L} = 2(\theta - y) = -2.3$$

$$\theta_{\text{new}} = 1 - 0.1(-2.3) = 1.23$$

Teaching assumption: one scalar parameter. A general neural network computes parameter gradients by the chain rule.

10.4.2 Deep Q-Networks: Replay and Frozen Targets

Replay Past Samples and Freeze the Target Network

Replay buffer

e_1 S → A	e_2 A → B	e_3 B → G
e_4 S → G	e_5 A → B	e_6 S → A

Completed 8 $k = 8$

Uniform draw order

1 5 4 3 3 6 1 5

Parameter snapshots

Online θ

S→A	A→B	B→G	S→G
1.503	3.241	4.360	1.200

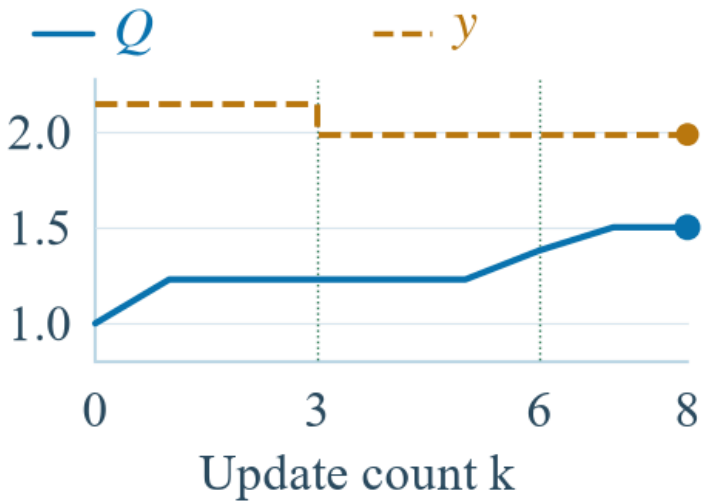
Target held θ^-

1.382 3.320 4.360 1.200

Copy after steps 3, 6

Step 6: y unchanged

Fixed probe S to A



$Q = 1.503$ $y = 1.988$

Linear Q, one sample per update: discount 0.9, learning rate 0.1; hard target copy every 3 updates

Initial Q-values for S-detour, A-forward, B→G, S-direct: 1, 3.5, 4, 1; G is terminal.

Six stored rewards: −1, −1, 5, 2, −1, −1. The blue line tracks only S-detour.

10.4.2 Deep Q-Networks: The Complete Algorithm

DQN Data Flow in Training and Execution

Initialize Two Networks
Copy Online to Target

→

Explore and Store
Sample Random Batches

→

Target y; Online Learns
Copy Target at Intervals

Stage	Required Components	Action Selection
Training	Online and target networks, replay buffer, environment	Online Q-values + exploration policy
Evaluation / Execution	Fixed online network, current state, control interface	Usually choose the action with the highest Q-value

Hard-copy the target at fixed intervals. Stop training using independent evaluation, budget and stability criteria.

10.4.2 Deep Q-Networks: Double, Dueling and Prioritized Replay

DQN Improvements Address Different Problems

Improvement	Problem	Mechanism
Double DQN	Maximization can overestimate values	Online network selects the next action; target network evaluates it
Dueling DQN	State quality and action differences are entangled	Combine a state-value branch and an action-advantage branch
Prioritized Experience Replay	Uniform sampling can be inefficient	Prioritize by factors such as absolute TD error; correct sampling bias

Advantage: action value relative to its state average. Large TD errors may also reflect noise.

Chapter Outline

Classic Single-Agent Reinforcement Learning Algorithms

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures

10.2 Consensus and Formation Control

10.3 Game Theory and RL Fundamentals

10.4 Single-Agent RL Algorithms

10.5 Multi-Agent RL and Scaling

10.6 Technologies and Challenges

10.7 RflySim Framework and Interfaces

10.8 Experiments and Validation

10.9 Exercises and Classroom Discussion

4 Units in This Section

10.4.1 Q-Learning

Textbook: Sec. 10.1.3.4, item 1

10.4.2 Deep Q-Networks

Textbook: Sec. 10.1.3.4, item 2

10.4.3 Actor–Critic and DDPG

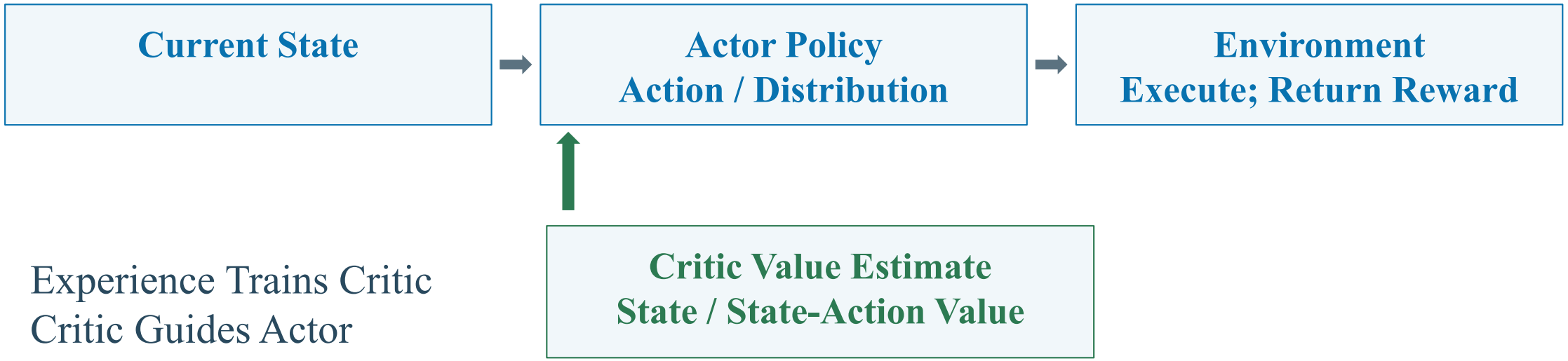
Textbook: Sec. 10.1.3.4, items 3–4

10.4.4 Algorithm Improvements and Selection

Textbook: Sec. 10.1.3.4, item 5

10.4.3 Actor–Critic and DDPG: The Actor–Critic Framework

The Actor Chooses; the Critic Evaluates



Experience Trains Critic
Critic Guides Actor

- **Actor:** change action selection to improve the long-term objective.
- **Critic:** learn values from rewards and later estimates to guide policy optimization.

10.4.3 Actor–Critic and DDPG: A2C and A3C

Advantage Measures an Action Against the Average

$$A_\mu(s,a) = Q_\mu(s,a) - V_\mu(s)$$

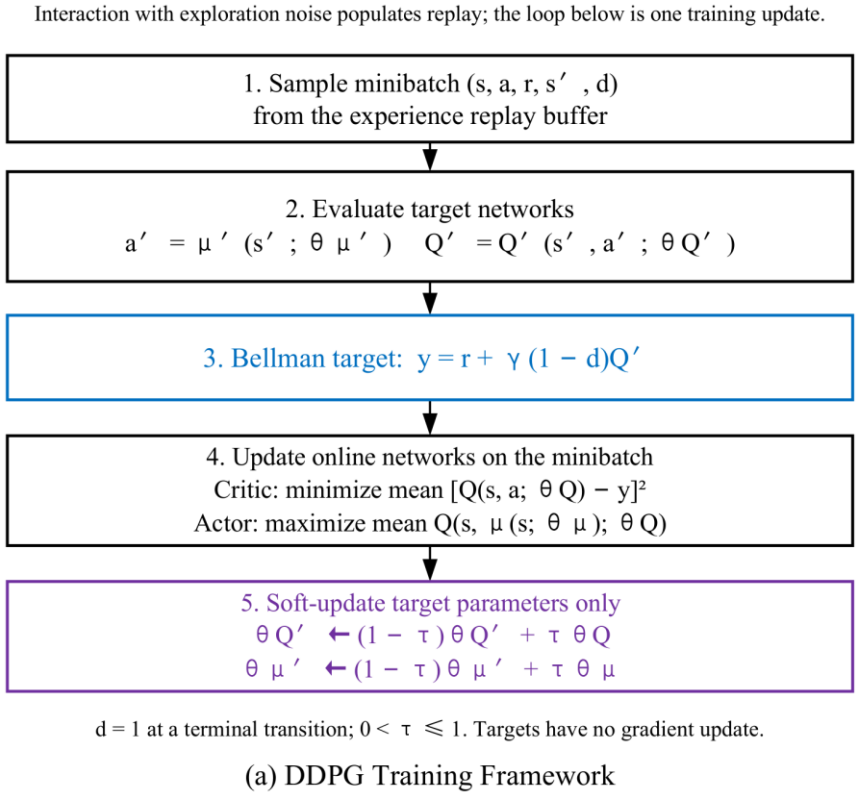
Equal probabilities for direct and detour: $V_\mu(S) = 0.5 \times 2 + 0.5 \times 2.15 = 2.075$

$$A_\mu(S, \text{via}) = 2.15 - 2.075 = 0.075$$

Method	Sampling and Updates	Shared Roles
Advantage Actor–Critic (A2C)	Parallel sampling; synchronous updates	Actor uses advantage to improve the policy
Asynchronous Advantage Actor–Critic (A3C)	Multiple workers update asynchronously	Critic estimates value to reduce gradient variance

10.4.3 Actor–Critic and DDPG: Fig. 10.7(a) Overview

DDPG Training Uses Four Networks



(a) DDPG Training Framework

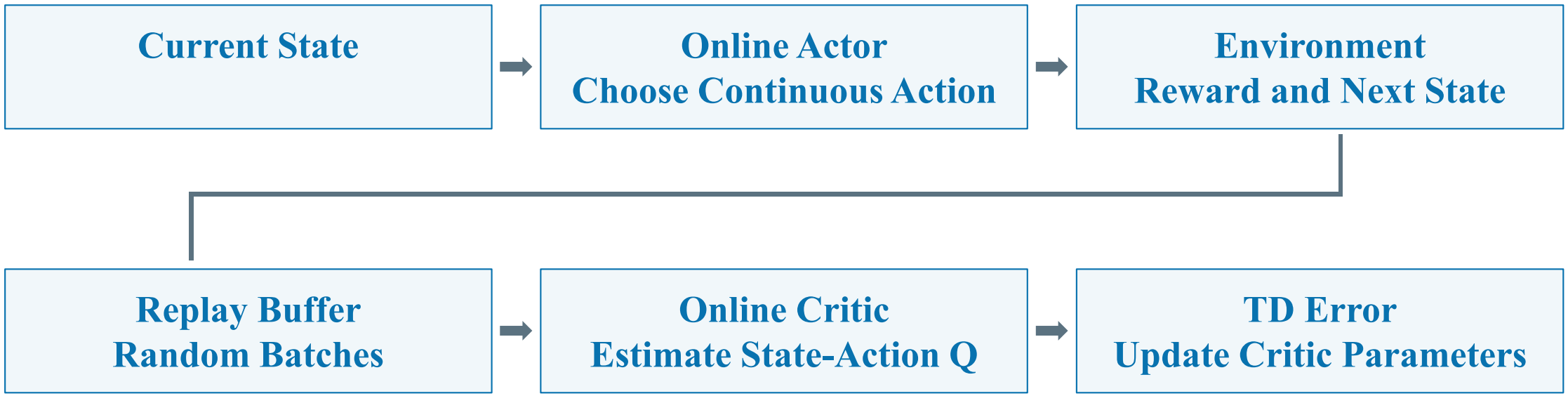
Fig. 10.7(a) DDPG training framework

- **Online actor:** map the state to a continuous action.
- **Online critic:** estimate the current state-action value.
- **Two target networks:** jointly construct the learning target.

Each target softly follows its own online network (Eq. 10.37); the two updates are separate.

10.4.3 Actor–Critic and DDPG: Fig. 10.7(a), Online Branch

Online Networks Collect Experience and Learn Values



The actor learns through the critic's action gradient; the critic learns from TD targets built from experience.

10.4.3 Actor–Critic and DDPG: Fig. 10.7(a), Target Branch

Target Networks Estimate the Next State's Value

Next State from Experience → **Target Actor Generate Next Action** → **Target Critic Estimate Next Action Value**

$$s' \longrightarrow \mu'(s') \longrightarrow Q'(s', \mu'(s'))$$

- **Inputs:** the target critic receives both the next state and the action generated by the target actor.
- **Updates:** Eq. 10.37 soft-updates each target from its own online network. Targets do not gradient-update one another.

10.4.3 Actor–Critic and DDPG: Eq. 10.35

The Actor Maximizes the Critic's Value Estimate

$$J(\theta^{\mu}) = \mathbb{E}_{s\sim D}[Q(s, \mu(s;\theta^{\mu});\theta^{Q})] \tag{10.35}$$

$$\nabla_{\theta^{\mu}} J \approx \mathbb{E}[\nabla_a Q(s,a)|_{a=\mu(s)} \nabla_{\theta^{\mu}} \mu(s)]$$

- **Direction:** ascend the Q gradient; an optimizer can instead minimize negative mean Q.
- **Fixed parameters:** hold the critic fixed but retain its action gradient so that gradients reach the actor.

10.4.3 Actor–Critic and DDPG: Eq. 10.36

The Critic Fits a TD Target from a Target Action

$$\mathcal{L}(\theta^Q) = \mathbb{E}_{(s,a,r,s')\sim D}[(y - Q(s,a;\theta^Q))^2] \tag{10.36}$$

$$y = r + \gamma Q'(s', \mu'(s'))$$

$$y = r + \gamma(1-d)Q'(s', \mu'(s'))$$

Line 2 is the textbook's nonterminal target; line 3 adds termination. Neither target network receives gradients in this step.

10.4.3 Actor–Critic and DDPG: Eq. 10.37

Soft Updates Make Targets Follow Slowly

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \tag{10.37}$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$$

$$\theta_{\text{target,new}} = 0.01 \times 2 + 0.99 \times 1 = 1.01$$

A smaller soft-update weight gives slower tracking. Each target follows its own online network.

10.4.3 Actor–Critic and DDPG: Eq. 10.38

Exploration and Evaluation Use Different Actions

$$a_t = \mu(s_t; \theta^{\mu}) + \mathcal{N}_t \tag{10.38}$$

$$a_t = \text{clip}(\mu(s_t) + \mathcal{N}_t, a_{\min}, a_{\max})$$

Stage	Action Generation	Purpose
Training	Policy output + noise, then clip	Explore and collect broader experience
Fixed-Policy Evaluation	Policy output with the same clipping	Measure the policy's own task performance

10.4.3 Actor–Critic and DDPG: A Scalar Example

Calculate Continuous-Action Improvement by Hand

$$Q(a) = 4 - (a - 0.5)^2, \quad a \in [-1,1], \quad a_0 = 0$$

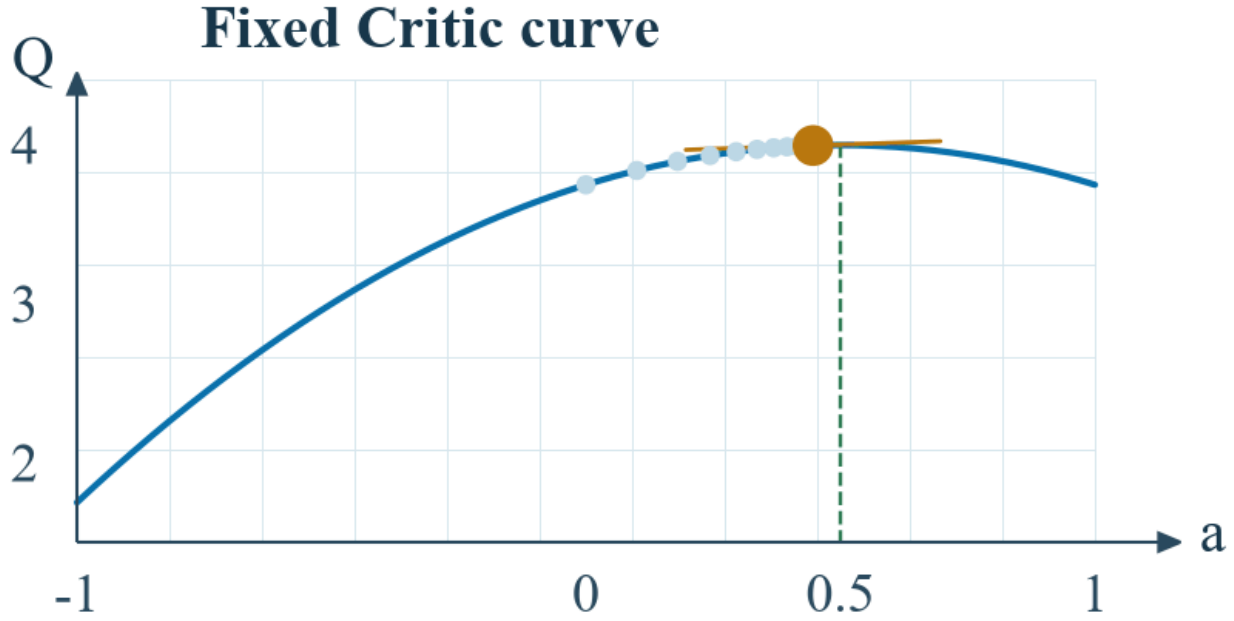
Fixed state and critic; the actor parameter equals the action. Learning rate = 0.1.

$$a_{k+1} = a_k + 0.1[-2(a_k - 0.5)] = 0.8a_k + 0.1$$

Updates	Action	Q from the Fixed Critic
0	0	3.7500
1	0.1000	3.8400
5	0.3362	3.9732
10	0.4463	3.9971

10.4.3 Actor–Critic and DDPG: The Action Gradient

The Actor Climbs the Critic's Value Curve



Parameter updates 10

Actor output 0.4463

Q(a) = 3.9971

dQ/da = 0.1074

a* = 0.5

Ten gradient-ascent steps with a fixed critic; the gold tangent shows the local slope

Near the peak, the slope and action updates shrink. This example updates only the actor.

10.4.3 Actor–Critic and DDPG: Sampling, Learning and Evaluation

The Complete DDPG Training Loop

Collect and Store
Keep Executed Actions → **Targets Compute y**
Update Online Critic → **Actor Maximizes Mean Q**
Soft-Update Both Targets

Step	Parameter Changes	Checks
Critic Update	Update only the online critic	TD error, termination mask, frozen target
Actor Update	Update only the online actor	Action gradient, clipping, Q bias
Independent Evaluation	Keep all policy parameters fixed	Return, success rate, constraint violations

Chapter Outline

Classic Single-Agent Reinforcement Learning Algorithms

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures

10.2 Consensus and Formation Control

10.3 Game Theory and RL Fundamentals

10.4 Single-Agent RL Algorithms

10.5 Multi-Agent RL and Scaling

10.6 Technologies and Challenges

10.7 RflySim Framework and Interfaces

10.8 Experiments and Validation

10.9 Exercises and Classroom Discussion

4 Units in This Section

10.4.1 Q-Learning

Textbook: Sec. 10.1.3.4, item 1

10.4.2 Deep Q-Networks

Textbook: Sec. 10.1.3.4, item 2

10.4.3 Actor–Critic and DDPG

Textbook: Sec. 10.1.3.4, items 3–4

10.4.4 Algorithm Improvements and Selection

Textbook: Sec. 10.1.3.4, item 5

10.4.4 Algorithm Improvements and Selection: Twin Critics and Delayed Updates

TD3: Twin Delayed Deep Deterministic Policy Gradient

- **Twin critics:** evaluate the same target action and take the smaller value, reducing overestimation the actor could exploit.
- **Delayed updates:** update the critics several times before the actor, so the policy uses more settled value estimates.
- **Target smoothing:** add clipped noise to the target action, then clip the action to reduce sensitivity to narrow, falsely high Q-peaks.

Policies exploit value errors; better evaluation improves decisions.

10.4.4 Algorithm Improvements and Selection: Stochastic Policies and Exploration

Soft Actor–Critic (SAC): Reward and Entropy

Stochastic Actor
Action Distribution
→
Reward + Policy Entropy
Return and Exploration
→
Critic Evaluation
Update Policy and Values

- **Entropy:** measure the randomness of the action distribution. More concentrated distributions usually have lower entropy.
- **Temperature:** weight the entropy term; adjust it toward a target entropy to control exploration.
- **Gradient:** express sampling as a differentiable combination of parameters and random noise so policy parameters receive gradients.

10.4.4 Algorithm Improvements and Selection: Probability-Ratio Clipping

PPO Clips the Objective to Discourage Large Updates

$$\rho_t = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\text{old}}(a_t \mid s_t)}$$

$$L^{\text{clip}} = \mathbb{E}[\min(\rho_t A_t, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)A_t)]$$

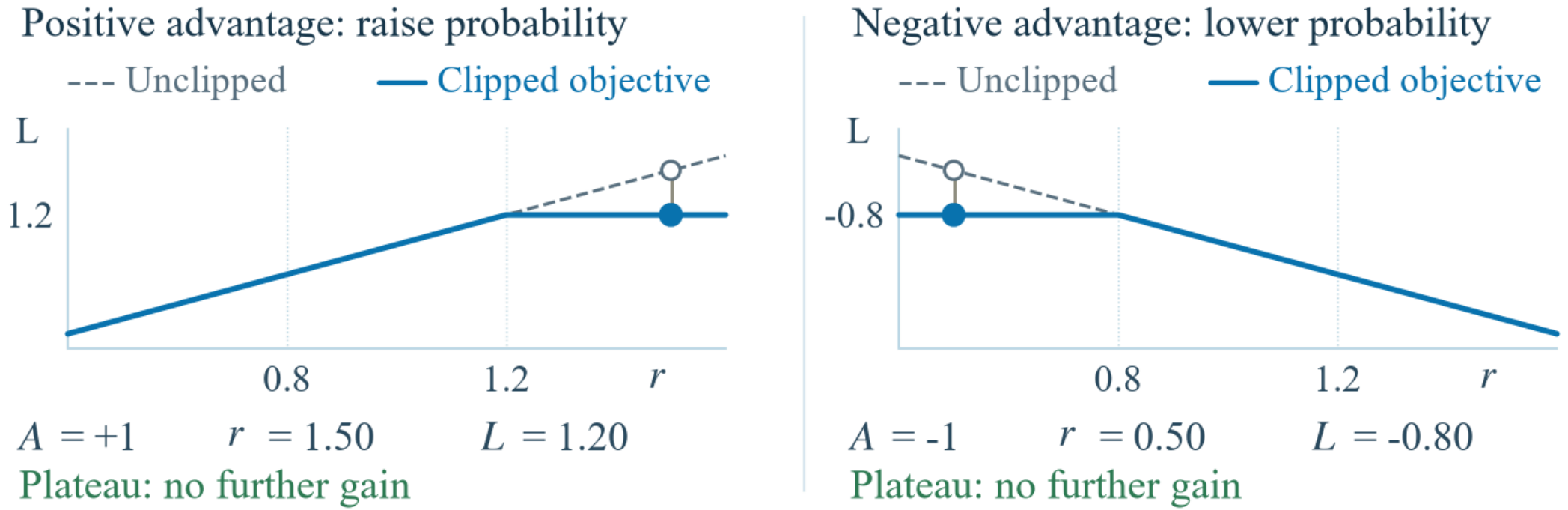
- Maximize the clipped objective; ρ_t = new/old action probability (density) ratio.
- A_t = old-policy advantage estimate; ϵ = clipping width.
- **Process:** generalized advantage estimation (GAE) weights multistep TD information to balance bias and variance. Update a batch a limited number of times, then resample.

$$A_t = 1, \quad \epsilon = 0.2, \quad \rho_t = 1.5: \quad \min(1.5, 1.2) = 1.2$$

For this positive-advantage sample, the objective term stops increasing above ratio 1.2.

10.4.4 Algorithm Improvements and Selection: A Single PPO Sample

Positive and Negative Advantages Flatten on Opposite Sides



r: previous-page probability ratio; clipping width 0.2. Filled points show the minimum.

Positive advantage flattens right; negative left. The actual ratio is not hard-clipped.

10.4.4 Algorithm Improvements and Selection: TD-MPC and TD-MPC2

TD-MPC Plans Short Trajectories in a Learned Model

Encode State
Enter Latent Space
→
Roll Out the Model
Compare Action Sequences
→
Reward + Terminal Value
Execute First Action

- TD-MPC (Temporal Difference Learning for Model Predictive Control): learn task dynamics, rewards and terminal values; plan in latent space.
- **TD-MPC2:** add a scalable world model, task embeddings, Q-value ensembles and value-scale handling.
- **Computation:** execution requires planning in the model. Model error and planning latency directly affect decisions.

10.4.4 Algorithm Improvements and Selection: Comparison

Choose by Actions, Data and Execution Budget

Method	Typical Action Space	Data and Execution
Q-Table / DQN	Finite Discrete	Off-policy; compare action Q-values at execution
DDPG / TD3	Continuous	Off-policy replay; deterministic actor forward pass
SAC	Usually Continuous	Off-policy replay; stochastic policy and entropy objective
PPO	Discrete or Continuous	Recent policy data; limited updates per batch
TD-MPC(2)	Continuous	Learn latent dynamics; plan online at execution

10.4.4 Algorithm Improvements and Selection: Materials and Task Scope

What the RflySim Single-Agent PPO Materials Cover

Materials (Full Edition or above; check your release package)
10.RflySimSwarm/3.CustExps/e0.CustApiExps/3.MARLExps

File	What to Check
Readme.pdf	Genesis + RflySim quadcopter acceleration-based hovering: task and dependencies
train/env_acc.py	Definitions of observations, actions, rewards and termination
train/train_acc.py	PPO training configuration and parallel sampling
sil/rfAcc.py	How fixed-policy outputs become software-in-the-loop (SITL; SIL in the textbook) control inputs

Parallel PPO environments collect single-agent experience; multi-agent tasks require interacting agents.

10.4.4 Algorithm Improvements and Selection: Further Study

Read the Principles and Implementations

DDPG objectives, termination and complete pseudocode
https://spinningup.openai.com/en/latest/algorithms/ddpg.html

PPO clipped objective and policy updates
https://spinningup.openai.com/en/latest/algorithms/ppo.html

TD-MPC2 papers, world model and implementation
https://www.tdmpc2.com/

Exercises and Discussion

Exercises: Targets and Network Roles

- **Q1:** Terminal reward = 5, old Q = 2, learning rate = 0.5. The next-state network outputs 100. What is the new Q?

- **Q2:** Online parameter = 2, target parameter = 1, soft-update weight = 0.1. What is the new target parameter?

- **Q3:** Must fixed-policy DDPG execution run the critic, target networks and replay buffer?

- **Q4:** One PPO policy samples 100 independent single-agent environments. Is this collaboration among 100 agents?

Reference Answers

Reference Answers: No Future Return After Termination

$$y = 5, \qquad Q_{\text{new}} = 2 + 0.5(5 - 2) = 3.5$$

$$\theta'_{\text{new}} = 0.1 \times 2 + 0.9 \times 1 = 1.1$$

- **Q3:** A fixed actor, observation preprocessing, action clipping and the interface support execution. The critic and related components usually serve training.
- **Q4:** This is parallel single-agent sampling. Multi-agent decisions interact within a shared environment.

Section Summary

Value Learning and Continuous-Action Optimization

- **Q-Learning and DQN:** update action values from samples. Shared network parameters couple updates, requiring careful data and target management.
- **Actor–Critic and DDPG:** the policy generates actions; the critic evaluates them. Value gradients can optimize continuous actions.
- **Selection:** compare action space, experience reuse, exploration and execution budget, then design the evaluation.
- **Common errors:** terminal targets have no later-value term; targets receive no loss gradients; evaluation parameters and exploration rules stay fixed.