From Bandits to PPO

Lecture Notes for a One-Hour Reinforcement Learning Prerequisite Seminar

Yuhan Chi
Fudan University

April 2026

Abstract

These notes are organized around a single pedagogical objective: to prepare the audience for a follow-up seminar on PPO-family large-language-model post-training methods such as RLHF, GRPO, and DAPO without spending time on application-specific engineering details. The right stopping point is therefore *PPO*, not reward-model training or Transformer internals. The cleanest thread is to separate two questions: *what object do we learn* and *how do we estimate its learning signal*. Along this thread, Monte Carlo and temporal-difference learning are first presented as two ways to estimate values; they then induce two tabular control branches, namely a pedagogically simplified Monte Carlo policy-iteration scheme and the TD-style family containing SARSA and Q-learning. Once tabular methods become inadequate in large or continuous problems, function approximation splits the story again: approximating a value function leads to DQN, while parameterizing the policy leads to policy gradients. At that point, the same MC-vs.-TD contrast reappears: policy parameterization plus Monte Carlo returns gives REINFORCE, whereas policy parameterization plus a learned critic gives actor–critic, GAE, and finally PPO.

Contents

1 Positioning and scope

These notes inherit the backbone of the handwritten draft and the preliminary typed version: begin from bandits, move through Markov decision processes, and end at PPO. The important adjustment is one of organization. Instead of presenting methods as a flat list, the seminar is structured around two recurring questions:

- what is the primary object we learn: a value function or a policy?
- how is the learning signal estimated: by Monte Carlo returns or by temporal-difference bootstrapping?

This simple map turns an otherwise crowded landscape into a coherent development:

$$\begin{aligned}\text{MC/TD as value estimators} &\to \text{tabular control} \to \text{function approximation} \\ &\to \text{REINFORCE / actor–critic / PPO.}\end{aligned}$$

If the next seminar is about GRPO or DAPO, then the present seminar should not stop at REINFORCE; it must continue until the audience understands why modern policy optimization uses baselines, advantage functions, actor–critic structure, GAE, and PPO's clipped objective [1, 7, 8].

Suggested one-hour oral allocation

Topic	Time	Oral emphasis
Bandits and gradient bandits	7 min	expand
Contextual bandits	4 min	brief but keep
MDPs and value functions	8 min	expand
Dynamic programming	4 min	compress
MC and TD as two estimators	6 min	expand
Tabular control: MC basic, SARSA, Q-learning	6 min	bridge section
Function approximation: DQN vs. policy parameterization	5 min	orient the fork
Policy gradient and REINFORCE	6 min	expand
Baseline, advantage, actor–critic, GAE	7 min	essential
PPO and handoff	7 min	endpoint

Core notation

Symbol	Meaning
S_t, A_t, R_{t+1}	state, action, and reward at time t
$\pi_\theta(a \mid s)$	stochastic policy with parameters θ
G_t	discounted return from time t onward
$V_\pi(s)$, $Q_\pi(s,a)$	state-value and action-value under policy π
δ_t	TD residual / TD error
$A_\pi(s,a)$	advantage function $Q_\pi(s,a) - V_\pi(s)$
$\hat{A}_t$	estimated advantage, often from GAE
$r_t(\theta)$	PPO importance ratio $\pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$

2 Multi-armed bandits

2.1 Exploration and exploitation

A multi-armed bandit is the simplest sequential decision problem. At each round t, the learner chooses an action A_t and observes an immediate reward R_t. There is no action-induced future state, so the difficulty is not delayed credit assignment but rather the exploration–exploitation dilemma: whether to exploit the action that currently looks best or to explore actions whose true quality is still uncertain [1].

The true action value is

$$q_*(a) = \mathbb{E}[R_t \,|\, A_t = a]. \tag{1}$$

A standard incremental estimator is

$$Q_{t+1}(a) = Q_t(a) + \alpha(R_t - Q_t(a)). \tag{2}$$

Classical heuristics such as ε-greedy and UCB are useful to mention briefly:

$$A_t = \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]. \tag{3}$$

But for this seminar the real bridge is *gradient bandits*.

2.2 Preferences and softmax policies

Instead of estimating values and then acting greedily, gradient bandits introduce preferences $H_t(a)$ and map them into a stochastic policy via softmax:

$$\pi_t(a) = \frac{\exp(H_t(a))}{\sum_b \exp(H_t(b))}. \tag{4}$$

The expected reward under the current policy is

$$J(H) = \mathbb{E}[R_t] = \sum_a \pi_H(a) q_*(a). \tag{5}$$

This is already the first appearance of direct policy optimization: the policy itself becomes the object of learning.

2.3 Why the preference update takes this form

The derivative of the softmax is

$$\frac{\partial \pi(a)}{\partial H(c)} = \pi(a)(\mathbf{1}\{a = c\} - \pi(c)). \tag{6}$$

Differentiating the objective with respect to $H(c)$ yields

$$\begin{aligned} \frac{\partial J}{\partial H(c)} &= \sum_a q_*(a) \frac{\partial \pi(a)}{\partial H(c)} && (7)\\ &= \sum_a q_*(a)\pi(a)(\mathbf{1}\{a = c\} - \pi(c)) && (8)\\ &= \pi(c) \left(q_*(c) - \sum_a \pi(a) q_*(a) \right). && (9) \end{aligned}$$

So the gradient compares the value of action c against the policy-average value. Replacing the unknown $q_*(c)$ by the sampled reward R_t and the policy-average value by a running baseline $\bar{R}_t$ gives the stochastic-gradient updates

$$\begin{aligned} H_{t+1}(A_t) &= H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t)), && (10)\\ H_{t+1}(a) &= H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a), \qquad a \neq A_t. && (11) \end{aligned}$$

Remark 1. *The baseline $\bar{R}_t$ does not change the expected gradient direction; it reduces variance. Conceptually, this is the bandit ancestor of the value-function baseline and, later, the advantage function.*

3 Contextual bandits

A contextual bandit augments the one-step problem with an observed context x. The best action is now conditional on the observed situation, so the policy becomes

$$\pi_\theta(a \mid x). \tag{12}$$

A natural objective is

$$J(\theta) = \mathbb{E}_{x \sim \mathcal{D},\, a \sim \pi_\theta(\cdot|x)}[r(x, a)]. \tag{13}$$

This is already close to supervised input-output mapping, except that the learner only receives feedback for the sampled action. The crucial limitation is that actions still do *not* alter future contexts. Therefore contextual bandits remain a one-step decision problem, not a full Markov decision process [1].

4 MDPs, trajectories, and value functions

4.1 From one-step choice to sequential choice

A discounted Markov decision process is a tuple

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma), \tag{14}$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, P the transition kernel, R the reward mechanism, and $\gamma \in [0, 1)$ the discount factor. At time t, the agent observes S_t, chooses A_t, receives reward R_{t+1}, and the environment transitions according to

$$p(s', r \mid s, a). \tag{15}$$

A trajectory is written as

$$\tau = (s_0, a_0, r_1, s_1, a_1, r_2, \dots). \tag{16}$$

The fundamental new phenomenon is delayed consequence: a locally attractive action can be globally harmful, and vice versa.

4.2 Return, state value, and action value

The discounted return from time t onward is

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{17}$$

Under a policy π, the state-value and action-value functions are

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s], \qquad Q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]. \tag{18}$$

The Bellman expectation equation for V_π is

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a)\big(r + \gamma V_\pi(s')\big). \tag{19}$$

4.3 Finite-state matrix form

In a finite discounted MDP, let P_π denote the transition matrix under policy π and r_π the expected one-step reward vector. Then

$$V_\pi = r_\pi + \gamma P_\pi V_\pi, \tag{20}$$

so equivalently

$$(I - \gamma P_\pi) V_\pi = r_\pi, \qquad V_\pi = (I - \gamma P_\pi)^{-1} r_\pi. \tag{21}$$

This matrix viewpoint matches the handwritten notes closely: policy evaluation is a linear system in finite state spaces. It is mathematically elegant, but for large problems it is usually computationally infeasible.

4.4 Bellman operators and contraction

Define the Bellman operator by

$$(T_\pi V)(s) = \sum_a \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a)\big(r + \gamma V(s')\big). \tag{22}$$

Proposition 1 (Contraction of the Bellman expectation operator)**.** *For any bounded value functions V and W,*

$$\|T_\pi V - T_\pi W\|_\infty \le \gamma \|V - W\|_\infty . \tag{23}$$

Hence T_π admits a unique fixed point, namely V_π.

Proof. For each state s,

$$\begin{aligned} \left|(T_\pi V)(s) - (T_\pi W)(s)\right| &= \left| \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\gamma(V(s') - W(s')) \right| \\ &\le \gamma \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a) \|V - W\|_\infty \\ &= \gamma \|V - W\|_\infty . \end{aligned}$$

Taking the supremum over s proves the claim. □

Remark 2. *The proof is short, but it is already enough: Bellman operators contract in the discounted tabular setting, which is why iterative evaluation converges.*

5 A compressed dynamic-programming detour

The Bellman optimality equation is

$$V_*(s) = \max_a \sum_{s',r} p(s', r \mid s, a)\big(r + \gamma V_*(s')\big). \tag{24}$$

It gives rise to two classical exact-control procedures.

Algorithm 1: Value iteration

```
Initialize a value function V
repeat
    for each state s do
        V(s) ← max_a Σ_{s',r} p(s', r | s, a)(r + γV(s'))
until V stabilizes
Extract a greedy policy from the final value function
```

Algorithm 2: Policy iteration

```
Initialize a policy π
repeat
    Evaluate V_π under the current policy
    Improve the policy by making it greedy with respect to V_π
until the policy no longer changes
```

These methods are conceptually important because they reveal the fixed-point structure of control. But for the present seminar they should be compressed: they assume the model is known and sweep over the whole state space, which is far from the data-driven policy-optimization setting that leads to PPO.

6 Monte Carlo and temporal-difference learning

Exact dynamic programming is model-based and rarely practical in large environments. Model-free methods learn from sampled trajectories instead [1]. The first key pedagogical point is that Monte Carlo and temporal-difference learning are, before anything else, *two ways to estimate values from experience*. They are not yet complete control algorithms by themselves.

6.1 Monte Carlo evaluation

Monte Carlo waits until a complete return is observed:

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big). \tag{25}$$

It is conceptually direct and uses a full sampled return, so it avoids bootstrapping bias. The cost is that the target can have large variance, especially when rewards are delayed or episodes are long.

6.2 TD(0) evaluation

Temporal-difference learning bootstraps from the next value estimate:

$$V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big). \tag{26}$$

The corresponding TD residual is

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t). \tag{27}$$

This local prediction error becomes the central building block in actor–critic methods and GAE.

6.3 n-step returns and lambda-return

TD(0) and Monte Carlo are best viewed as endpoints of a continuum. The n-step return is

$$G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k R_{t+k+1} + \gamma^n V(S_{t+n}). \tag{28}$$

Special cases are immediate:

- $n = 1$ gives the TD(0) target;
- taking n to the episode end gives the Monte Carlo return.

The λ-return geometrically averages these horizons:

$$G_t^{\lambda} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}, \qquad 0 \le \lambda \le 1. \tag{29}$$

This spectrum from short-horizon bootstrapping to long-horizon Monte Carlo is exactly the conceptual bridge needed before GAE.

6.4 Bias–variance interpretation

Monte Carlo and TD are best compared through the bias–variance tradeoff. Monte Carlo uses the complete sampled return and therefore avoids bootstrapping bias, but this long-horizon target can have substantial variance. TD, by contrast, replaces the full return with a bootstrapped target built from the current value estimate. That substitution often reduces variance dramatically, but it introduces bias because the target itself depends on an estimate that is still changing.

It is important to phrase this carefully: TD is not biased because one has "introduced the Markov assumption." The Markov property is part of the modeling framework for the environment. The statistical bias comes specifically from bootstrapping, that is, from replacing a full realized return by a target that already contains an estimated value.

7 From value estimation to tabular control

Once a way of estimating values has been chosen, control is obtained by coupling evaluation with policy improvement. This is the point at which the lecture can naturally branch into two very simple tabular families.

7.1 MC basic control: policy evaluation plus greedy improvement

In these notes, by *MC basic* we mean the pedagogically simplest Monte Carlo control scheme: fix a policy π, run it for many complete episodes, estimate $q_\pi(s,a)$ by averaging observed returns, and then improve the policy greedily with respect to those estimates. Conceptually, this is a weakened classroom version of Monte Carlo exploring starts or ε-soft Monte Carlo control. What is omitted is a formal exploration guarantee; what is retained is the core idea that Monte Carlo can serve as the evaluation step inside policy iteration.

If a state-action pair (s,a) is visited at time t, one may estimate

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \tag{30}$$

by sample averaging, and then improve the policy through

$$\pi_{\text{new}}(s) \in \arg\max_a \hat{q}_\pi(s,a). \tag{31}$$

Algorithm 3: MC Basic Iteration

Initialize a policy π and an exploratory behavior rule
repeat
  Roll out many complete episodes under π or an exploratory version of π
  for each *visited state-action pair* (s,a) **do**
    Estimate $\hat{q}_\pi(s,a)$ by averaging realized returns following occurrences of (s,a)
  Improve the policy by setting $\pi(s) \leftarrow \arg\max_a \hat{q}_\pi(s,a)$ on states with enough data
until *the policy stabilizes or the budget is exhausted*

The point of keeping this branch in the lecture is not that it is the final modern algorithm. Rather, it shows very clearly what Monte Carlo means when turned into control: *evaluate from complete returns, then improve greedily.*

7.2 SARSA and Q-learning: the TD control branch

The TD side yields control methods that bootstrap instead of waiting for complete returns. The canonical on-policy update is SARSA:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big). \tag{32}$$

Because the target uses the next action A_{t+1} that was actually sampled from the current behavior policy, SARSA is naturally on-policy.

The canonical off-policy update is Q-learning:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\Big). \tag{33}$$

Here the target uses the greedy next action whether or not that action was actually taken, so the method is off-policy [3].

Method	Target form	Interpretation
MC Basic Iteration	full sampled return	evaluation by complete episodes, then greedy improvement
SARSA	$r + \gamma Q(s', a')$	TD control using the action actually sampled next
Q-learning	$r + \gamma \max_{a'} Q(s', a')$	TD control aiming directly at the greedy target

8 Why tabular control eventually becomes insufficient

At this stage the lecture should explicitly explain what breaks. The issue is not that MC, SARSA, or Q-learning become mathematically meaningless as soon as the state space is "infinite." The issue is that their *tabular form* stops being practical or even representable in large or continuous problems.

Three distinct difficulties should be separated.

- **Representation:** a table cannot scale to huge or continuous state spaces, and it cannot generalize from one state to nearby states.
- **Estimator quality:** Monte Carlo targets can have very large variance, while TD targets trade variance for bootstrapping bias.
- **Greedy maximization:** the $\max_{a'}$ in Q-learning is convenient for small discrete actions, but it becomes difficult in continuous action spaces and can also introduce overestimation bias.

These are the real reasons for introducing function approximation.

9 Function approximation: two directions

Once tables are no longer viable, there are two natural choices.

- Approximate a **value function** such as $V_\phi(s)$ or $Q_\phi(s, a)$.
- Parameterize the **policy itself** as $\pi_\theta(a \mid s)$.

This fork is one of the most important structural decisions in reinforcement learning.

9.1 Approximating a value function: DQN

When the state space is too large for a table, one may replace Q by a parameterized approximator $Q_\theta(s, a)$. In this sense, DQN is best understood as *Q-learning plus function approximation* together with several stabilizing devices. A standard DQN objective is

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right], \tag{34}$$

where θ^- denotes a delayed target network for stability [5]. Experience replay and target networks are implementation details worth mentioning briefly, but the conceptual slogan is simple: *DQN is the value-based, TD-style, function-approximation branch.*

9.2 Approximating the policy: the door to policy gradients

The alternative is to represent the policy directly by parameters θ. This is especially natural when the action space is continuous or when one wishes to optimize stochastic action probabilities directly. Once the policy is parameterized, the old MC-vs.-TD contrast reappears in a new form:

Primary object	Monte Carlo style signal	TD / bootstrapped signal
Value function	MC basic control	SARSA, Q-learning, DQN
Policy	REINFORCE	actor–critic, GAE, PPO

This table is the central navigation map for the rest of the seminar. REINFORCE is what one gets when policy parameterization is combined with Monte Carlo returns; actor–critic is what one gets when policy parameterization is combined with a learned value estimator.

10 What to optimize: value-based versus policy-based methods

At this point the strategic fork becomes explicit:

- **Value-based methods** learn V or Q first and then derive a policy.
- **Policy-based methods** optimize the policy parameters directly.

Since PPO and its descendants are policy-optimization methods, we now switch decisively to the policy-based thread.

11 Policy gradient and REINFORCE

If the policy itself is parameterized, there are again two choices for constructing the learning signal. The Monte Carlo branch gives REINFORCE; the bootstrapped branch, developed later, gives actor–critic. We start with the Monte Carlo side because it is algebraically the cleanest.

11.1 Trajectory objective

Let $G(\tau)$ denote the return of a trajectory τ. The policy objective is

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}\big[G(\tau)\big]. \tag{35}$$

The trajectory distribution factorizes as

$$p_\theta(\tau) = \rho(s_0) \prod_{t=0}^{T-1} p(s_{t+1} \mid s_t, a_t)\, \pi_\theta(a_t \mid s_t), \tag{36}$$

where ρ is the initial-state distribution.

11.2 Score-function identity and the policy-gradient estimator

The crucial algebraic step is the score-function identity

$$\nabla_\theta p_\theta(\tau) = p_\theta(\tau) \nabla_\theta \log p_\theta(\tau). \tag{37}$$

Therefore

$$\nabla_\theta J(\theta) = \nabla_\theta \int G(\tau) p_\theta(\tau)\, \mathrm{d}\tau \tag{38}$$

$$= \int G(\tau) \nabla_\theta p_\theta(\tau)\, \mathrm{d}\tau \tag{39}$$

$$= \mathbb{E}_{\tau \sim p_\theta}\big[G(\tau) \nabla_\theta \log p_\theta(\tau)\big]. \tag{40}$$

Since the environment dynamics do not depend on θ,

$$\nabla_\theta \log p_\theta(\tau) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t). \tag{41}$$

Substituting this factorization gives

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta}\left[\sum_{t=0}^{T-1} G(\tau) \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]. \tag{42}$$

Using the causality refinement, one may replace the trajectory-level return $G(\tau)$ by the reward-to-go G_t, which yields the REINFORCE estimator [2]:

$$\hat{g} = \sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, G_t. \tag{43}$$

11.3 Alternative Q-pi form

In expectation, the same gradient can be written in the standard policy-gradient-theorem form

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_t \sim d_{\pi_\theta},\, a_t \sim \pi_\theta(\cdot|s_t)}\big[Q^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t \mid s_t)\big], \tag{44}$$

where d_{π_θ} denotes the discounted state visitation distribution. If one suppresses this distributional notation, the expression above is the compact form that underlies the policy-gradient theorem [4].

Remark 3. *REINFORCE is mathematically clean, but its Monte Carlo estimator can have prohibitively high variance. Modern deep reinforcement learning therefore almost never stops here.*

Algorithm 4: REINFORCE

```
Initialize policy parameters θ
repeat
    Sample trajectories τ under π_θ
    Compute the reward-to-go G_t for each time step
    θ ← θ + α Σ_t ∇_θ log π_θ(a_t | s_t) G_t
until a stopping criterion is met
```

12 Baselines, advantage functions, and actor–critic

12.1 Why subtracting a baseline helps

Consider the modified estimator

$$\hat{g} = \sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)(G_t - b(s_t)). \tag{45}$$

Any baseline that depends only on the state leaves the expected gradient unchanged.

Proposition 2 (Baseline invariance)**.** *For any function $b(s)$,*

$$\mathbb{E}_{a \sim \pi_\theta(\cdot|s)}\big[b(s) \nabla_\theta \log \pi_\theta(a \mid s)\big] = 0. \tag{46}$$

Proof. Because $b(s)$ is independent of the sampled action,

$$\begin{aligned}
\mathbb{E}_{a \sim \pi_\theta(\cdot|s)}\big[b(s) \nabla_\theta \log \pi_\theta(a \mid s)\big] &= b(s) \sum_a \pi_\theta(a \mid s) \nabla_\theta \log \pi_\theta(a \mid s) \\
&= b(s) \sum_a \nabla_\theta \pi_\theta(a \mid s) \\
&= b(s) \nabla_\theta \sum_a \pi_\theta(a \mid s) \\
&= b(s) \nabla_\theta 1 = 0.
\end{aligned}$$

□

So subtracting a baseline does not change the expected gradient direction, but can reduce variance substantially.

12.2 Advantage functions

The most natural state-dependent baseline is the state value itself. This leads to the advantage function

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s). \tag{47}$$

It measures how much better or worse an action is relative to the average value level of its state. Writing the policy gradient in terms of A_π makes the estimation problem more localized:

$$\nabla_\theta J(\theta) = \mathbb{E}\big[A_\pi(S_t, A_t)\nabla_\theta \log \pi_\theta(A_t \mid S_t)\big]. \tag{48}$$

12.3 Actor–critic

The clean conceptual slogan is: *REINFORCE is the Monte Carlo policy-gradient method, whereas actor–critic is the TD policy-gradient method.* Actor–critic methods split the problem into two coupled learners:

- the **actor**, which updates the policy π_θ;
- the **critic**, which estimates a value function V_ϕ.

The simplest one-step actor–critic uses the TD residual

$$\delta_t = R_{t+1} + \gamma V_\phi(S_{t+1}) - V_\phi(S_t) \tag{49}$$

as an estimator of the advantage. A canonical pair of updates is

$$\theta \leftarrow \theta + \alpha_\pi\, \delta_t\, \nabla_\theta \log \pi_\theta(A_t \mid S_t), \tag{50}$$

$$\phi \leftarrow \phi - \alpha_v \nabla_\phi \big(V_\phi(S_t) - \hat{V}_t\big)^2. \tag{51}$$

The critic loss $\big(V_\phi(S_t) - \hat{V}_t\big)^2$ raises a natural question: what should the target $\hat{V}_t$ be? There are three standard choices, ranging from pure bootstrapping to pure Monte Carlo.

Choice 1: One-step TD target (bootstrapped)

This is the standard choice for one-step actor–critic methods such as the basic form of A2C:

$$\hat{V}_t = R_{t+1} + \gamma\, V_\phi(S_{t+1}). \tag{52}$$

The idea is that although the true value of S_t is unknown, we can take one step in the environment, observe the real immediate reward R_{t+1}, and then add the discounted value estimate of the next state. This combination is a better proxy for the true value than the current estimate $V_\phi(S_t)$ alone. Because the target depends on the parameterized V_ϕ, this is a bootstrapped estimate: it is low in variance but introduces bias.

Choice 2: Monte Carlo return

If the critic is trained offline after an episode terminates, one can use the full sampled return:

$$\hat{V}_t = G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{53}$$

Here G_t is the real sum of discounted rewards collected from S_t until the end of the episode. This target has zero bias but typically very high variance, especially in long episodes with delayed rewards.

Choice 3: n-step TD / GAE target

In modern deep reinforcement learning algorithms such as PPO and TRPO, the standard practice is to balance bias and variance by using multi-step returns or the λ-return. In this context, the λ-return is best viewed as a common target for critic training or as the building block of advantage estimation. Recall from section 6 the λ-return:

$$\hat{V}_t = G_t^{\lambda} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}, \tag{54}$$

where $G_t^{(n)}$ is the n-step truncated return plus the bootstrapped value at step n. When $\lambda = 0$ the target reduces to the one-step TD target (low variance, high bias); when $\lambda \approx 1$ it approaches the full Monte Carlo return (low bias, high variance). In practice, λ is a hyperparameter that the practitioner tunes to find the best tradeoff for the task at hand.

Algorithm 5: One-step actor–critic

Initialize policy parameters θ and critic parameters ϕ
repeat
  Sample $(S_t, A_t, R_{t+1}, S_{t+1})$ under π_θ
  $\delta_t \leftarrow R_{t+1} + \gamma V_\phi(S_{t+1}) - V_\phi(S_t)$
  $\theta \leftarrow \theta + \alpha_\pi \, \delta_t \, \nabla_\theta \log \pi_\theta(A_t \mid S_t)$
  Update the critic by descending the squared TD error or value-loss objective
until *a stopping criterion is met*

13 Generalized advantage estimation

GAE is the standard modern answer to the question: how should one estimate advantages well enough for stable policy optimization? Define the TD residual again as

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t). \tag{55}$$

Then generalized advantage estimation is

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}. \tag{56}$$

This estimator also admits the equivalent relation

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = G_t^{\lambda} - V(s_t), \tag{57}$$

which makes its connection to the λ-return explicit [7].

The key extremes are worth memorizing:

- $\lambda = 0$: mostly one-step TD, lower variance, more bias;
- $\lambda \approx 1$: closer to Monte Carlo, lower bias, higher variance.

Remark 4. *GAE is* not *a new optimization objective. It is an advantage estimator. This distinction matters because in PPO the objective is the clipped surrogate, whereas GAE provides the $\hat{A}_t$ that feeds that objective.*

14 PPO: proximal policy optimization

14.1 Why plain policy gradient is unstable

Policy-gradient updates are sensitive to step size. A small movement in parameter space need not correspond to a small movement in action-probability space, and large policy shifts can rapidly invalidate the data used for estimation. TRPO addresses this with a trust-region constraint [6], but its second-order machinery is relatively heavy. PPO keeps the spirit of constrained improvement while remaining first-order and simple to implement [8].

14.2 Importance ratios and surrogate objectives

Given samples generated by an old policy $\pi_{\theta_{\mathrm{old}}}$, define the probability ratio

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t \mid s_t)}. \tag{58}$$

If $r_t(\theta) = 1$, the new policy agrees with the old one on the sampled action. Deviations away from 1 measure how much the action probability has changed.

The unclipped surrogate would maximize $r_t(\theta)\hat{A}_t$, but PPO instead uses the clipped objective

$$L^{\mathrm{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min\Big(r_t(\theta)\hat{A}_t,\ \mathrm{clip}\big(r_t(\theta), 1-\varepsilon, 1+\varepsilon\big)\hat{A}_t \Big) \right]. \tag{59}$$

A common practical objective augments this with a value loss and an entropy bonus:

$$L^{\mathrm{PPO}}(\theta) = L^{\mathrm{CLIP}}(\theta) - c_v L^{\mathrm{VF}}(\theta) + c_e\, \mathcal{H}[\pi_\theta]. \tag{60}$$

14.3 Why clipping works

The clipping mechanism is easiest to understand case by case.

- If $\hat{A}_t > 0$, the sampled action is better than expected, so we want to increase its probability. But once $r_t(\theta)$ exceeds $1 + \varepsilon$, clipping stops rewarding further increase.
- If $\hat{A}_t < 0$, the sampled action is worse than expected, so we want to decrease its probability. But once $r_t(\theta)$ falls below $1 - \varepsilon$, clipping stops rewarding further decrease.

Hence PPO is well summarized as *policy gradient with a safety belt*. It allows multiple minibatch passes over the same batch while discouraging destructive policy drift.

14.4 The practical PPO loop

Algorithm 6: PPO

Initialize policy parameters θ, critic parameters ϕ, and set $\theta_{\text{old}} \leftarrow \theta$
repeat
  Collect trajectories under $\pi_{\theta_{\text{old}}}$
  Fit or update the critic V_ϕ
  Estimate advantages $\hat{A}_t$, typically with GAE
  for *several epochs of minibatch optimization* **do**
    Update θ by ascending L^{CLIP}
    Update ϕ with a value-loss objective
  Set $\theta_{\text{old}} \leftarrow \theta$
until *a stopping criterion is met*

15 Why PPO is the right stopping point before GRPO

At the end of these notes the audience should already understand

- why policy gradients contain $\nabla \log \pi$,
- why raw Monte Carlo returns are often too noisy,
- why baselines and advantage functions reduce variance,
- why actor–critic is a practical architecture,
- why GAE is the standard modern advantage estimator,
- and why PPO stabilizes policy updates with a ratio-based clipping mechanism.

This is precisely the level of reinforcement-learning background that makes the next seminar natural. GRPO can then be introduced as a PPO-family variant that modifies how the policy-improvement signal is normalized or baseline-corrected, and DAPO as a further refinement for stable reasoning-oriented policy optimization.

Acknowledgment of source material

The thread of these notes follows the user's handwritten pages on bandits, contextual bandits, and MDP matrix forms, together with the preliminary typed draft that already arranged the seminar into the sequence "Bandits → Contextual Bandits → MDPs → MC/TD → Policy Gradient → GAE → PPO." The present version mainly tightens the exposition, formalizes a few proofs, adds algorithm boxes, and supplies references.

I would also like to explicitly acknowledge Shiyu Zhao's book *Mathematical Foundations of Reinforcement Learning* as my main self-study text for reinforcement learning. In a very real sense, the overall organization and much of my understanding were learned from systematically studying that book, so these notes are written with substantial borrowing of perspective and exposition from it, even when the presentation here is shortened and adapted to the needs of this seminar [9].

I am also grateful to the Zhihu platform (https://www.zhihu.com) and its community of contributors. Many insightful discussions, tutorials, and expository articles on Zhihu have greatly enriched my understanding of reinforcement learning and related topics, and I have benefited enormously from browsing and learning from the wealth of knowledge shared there.

References

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd edition, MIT Press, 2018. http://incompleteideas.net/book/the-book-2nd.html.

[2] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.

[3] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3–4):279–292, 1992.

[4] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057–1063, 2000.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[6] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1889–1897, 2015.

[7] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations*, 2016.

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint* arXiv:1707.06347, 2017.

[9] Shiyu Zhao. *Mathematical Foundations of Reinforcement Learning*. Springer Press, 2025.