

Policy-Based Reinforcement Learning

Optimal control with the dynamics replaced by data

Jinkyoo Park · KAIST

Lecture 9 derived feedback from a model; this lecture learns a feedback policy from experience.

Where we are — the control lineage, made data-driven

PREVIOUS STEP	THIS CHAPTER'S QUESTION
Lecture 9 computed feedback from a specified dynamics model.	Now improve a parameterized policy using experience.

Values and policies work together in actor–critic methods. A learned actor supplies actions; its training objective and data determine how it improves.

Compute a policy-gradient step, a TD3 target and an entropy-regularized choice.

Learning route — make a good action more likely

Bring: Log derivatives, expectations, Bellman targets and feedback.

FIRST PASS	WHAT TO DO
Follow the idea	REINFORCE → baseline/critic → DDPG/TD3 → SAC → compare PPO
Work without the solution	Compute a policy-gradient step, a TD3 target and an entropy-regularized choice.
Return later	Full policy-gradient/Fisher derivations, GAE details and multi-agent extensions are references.

For the temperature thread: **predict** → **calculate** → **reveal and check** → **change one condition**. Complete the core calculation before reading the research extensions.

— ACT 1

A gradient without the model

Q1. The thing we must differentiate is an expectation over trajectories, and the dynamics are inside it.

The objective, and a gradient we cannot see

Q1 Gradient?

Q2 Variance?

Q3 Continuous?

Q4 Safe step?

Parameterise a stochastic policy $\pi_\theta(a \mid s)$ and maximise the expected return:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} [R(\tau)], \quad \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

The trouble is where the parameter lives. It is not in the integrand; it is in the **distribution being integrated against**, and that distribution contains the dynamics:

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

THE NAIVE ANSWER, AND WHY IT IS NOT ENOUGH

the source deck's own first attempt

Perturb one coordinate at a time and difference: $\partial J / \partial \theta_k \approx (J(\theta + \epsilon u_k) - J(\theta)) / \epsilon$. It can estimate local changes even without an analytic derivative — and it costs **one full batch of rollouts per parameter**. For a large network, coordinatewise finite differences are prohibitively expensive; random-direction methods offer a different trade-off.

The score-function trick — the dynamics vanish

One identity does all the work: $\nabla p = p \nabla \log p$. It turns the gradient of an expectation into an expectation of a gradient:

$$\nabla_{\theta} \mathbb{E}_{\tau \sim p_{\theta}} [R(\tau)] = \int \nabla_{\theta} p_{\theta}(\tau) R(\tau) d\tau = \int p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) R(\tau) d\tau = \mathbb{E}_{\tau \sim p_{\theta}} [\nabla_{\theta} \log p_{\theta}(\tau) R(\tau)]$$

Now expand that log. The product becomes a sum, and every term without θ differentiates to zero:

$$\nabla_{\theta} \log p_{\theta}(\tau) = \underbrace{\nabla_{\theta} \log \mu(s_0)}_{=0} + \sum_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) + \underbrace{\sum_t \nabla_{\theta} \log P(s_{t+1} | s_t, a_t)}_{=0} = \sum_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$

Their log-density derivatives are zero when they do not directly depend on θ . They still affect which trajectories occur and therefore the expected gradient.

This is the precise sense in which f is "deleted": we never needed it. Where Lecture 9 *solved* the dynamics, here they simply drop out of the algebra — the policy's own log-probability is all that is left. We still *live* in the dynamics; we just never differentiate through them.

Reading the estimator — it is weighted maximum likelihood

Put the two halves together and the whole of Act 1 is one line:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\underbrace{\left(\sum_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right)}_{\nabla_{\theta} J_{\text{ML}}(\theta) - \text{make this trajectory likelier}} \underbrace{\left(\sum_t r_{t+1} \right)}_{\text{how good it was}} \right]$$

The first factor, alone, is the gradient of the log-likelihood of the trajectory — [the maximum-likelihood gradient of Lecture 2](#), the direction that would make *this* trajectory more probable. The return simply weights it.

- **Increase** the likelihood of trajectories with large accumulated reward;
- **decrease** likelihood for negative weights. A small positive return still has a positive weight; a baseline lets us compare with typical performance.

Policy gradient is [imitation of your own better moments](#), reweighted by how well they went.

There is no `arg max` anywhere in that sentence, and no model. It is also, exactly, Lecture 6's move — a generative model trained to emit the *good* samples — one axis over.

The policy gradient theorem, and REINFORCE

For $J(\theta) = \mathbb{E}[\sum_{t=0}^{T-1} \gamma^t r_{t+1}]$, define $G_t = \sum_{k=t}^{T-1} \gamma^{k-t} r_{k+1}$. Then

$$\nabla_{\theta} J = \mathbb{E} \left[\sum_{t=0}^{T-1} \gamma^t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) G_t \right].$$

REINFORCE: sample complete episodes under the current policy, average this expression, and take a gradient-ascent step. The full-return estimator is unbiased under the usual differentiation and integrability assumptions.

Use **one score × its future return per action**. For the undiscounted episodic examples below, $\gamma = 1$. Dropping γ^t while claiming the same discounted objective is generally incorrect.

The equivalent Q^{π} theorem averages over the policy's state visitation weights. The appendix makes those weights explicit.

One policy parameter — see the update direction

A one-step problem has actions A and B. Let $\pi_{\theta}(A) = p = \text{sigmoid}(\theta) = 0.5$. Then the log-probability derivatives are $1 - p = 0.5$ for A and $-p = -0.5$ for B.

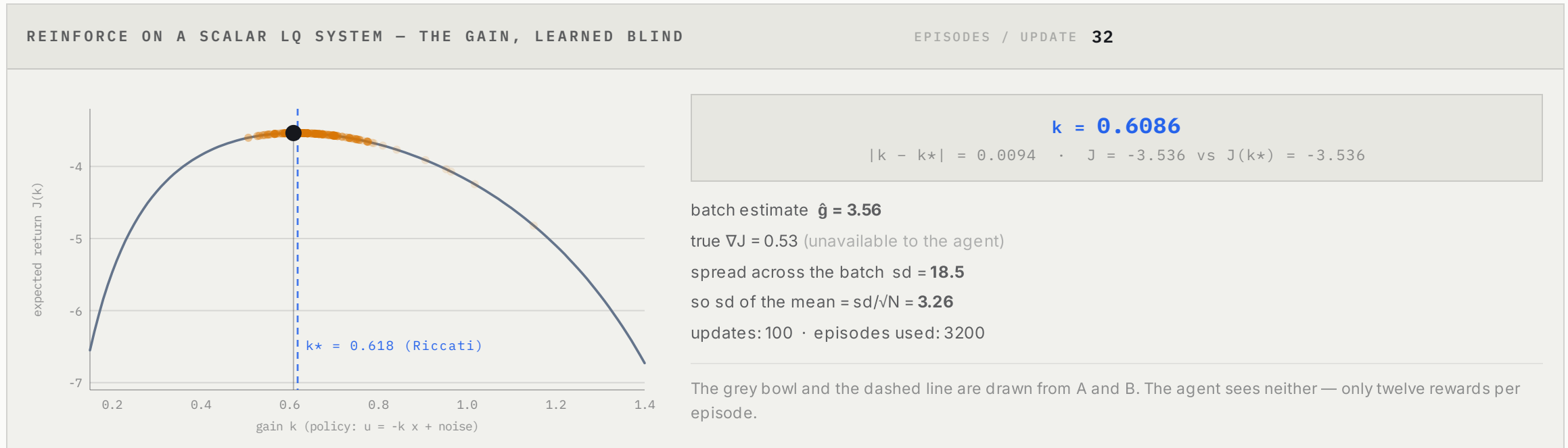
With step size 0.1 and baseline 2:

OBSERVED ACTION	REWARD	ADVANTAGE	GRADIENT CONTRIBUTION	PARAMETER CHANGE
A	3	1	$0.5(1) = 0.5$	+0.05
B	1	-1	$(-0.5)(-1) = 0.5$	+0.05

Both observations increase the probability of A: A performed above the baseline; B performed below it. **The sign comes from both the score and the advantage.**

REINFORCE, run — and what it converges to

a scalar linear system, learned blind



$x_{t+1} = x_t + u_t$ with cost $\sum (x^2 + u^2)$ — Lecture 9's problem, scalar. The policy is $u = -kx + \sigma\epsilon$ and the only parameter is the gain k . REINFORCE never sees A or B ; it only samples. The grey bowl $J(k)$ and the dashed line $k^* = 0.618$, the Riccati gain, are drawn for us , not for the agent — and that is where the gain walks to. Read the two numbers on the right: the estimate scatters *around* the true gradient, never away from it, and its spread falls like $1/\sqrt{N}$. Then set the batch to 8 and watch the gain stop settling — **that residual wander is Act 2's subject.**

Check — why differentiate the policy at all

Value-based RL picks actions by $\arg \max_a Q(s, a)$. Why does that break down in continuous control?

- ☒ A Because the $\arg \max$ is itself an optimisation over a continuous space, solved afresh at every single step
- ☐ B Because Q cannot be represented for continuous actions
- ☐ C Because the Bellman equation does not hold in continuous action spaces
- ☐ D Because rewards become unbounded

A table lookup over four actions is free; a maximisation over a continuous vector is an inner optimisation problem, and it recurs at every timestep of every episode. Policy-based methods sidestep it by **storing the answer**: a parameterised $\pi_\theta(s)$ that outputs the action directly. That is the same forward/inverse move as Lectures 5 and 6, now in a sequential setting.

— ACT 2

Taming variance with a critic

Q2. How can we reduce noise? Reward-to-go and action-independent baselines preserve the expected gradient; bootstrapping with an approximate critic can introduce bias.

Why the estimate is so noisy



Look again at what one episode contributes:

$$\hat{g}^{(i)} = \left(\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right) \left(\sum_{t=0}^{T-1} r_{t+1} \right)$$

Every score term is multiplied by the **same** scalar — the whole episode's return. One unlucky tail rescales all T of them together. And the return itself is a sum of T random rewards along a random trajectory.

Worse, it is *credit assignment by superstition*: an action at $t = 40$ is given credit for a reward already received at $t = 1$, which it could not have caused.

Three corrections follow, and **the first two preserve the mean; a bootstrapped critic adds an accuracy–variance trade-off.**

Keep the distinction clear: subtracting a fixed state baseline from a Monte Carlo return is not the same operation as replacing the return with a learned TD target.

Fix 1— causality: half the terms were pure noise

An action at time t cannot affect a reward at time $t' < t$. So those cross-terms have mean zero and contribute **nothing but variance**. Drop them:

$$\nabla_{\theta} J = \mathbb{E} \left[\sum_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{k=t}^{T-1} r_{k+1} \right] = \mathbb{E} \left[\sum_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \hat{Q}_t \right]$$

For an on-policy rollout continued to termination (or including the full discounted future), the inner sum is the **reward-to-go** $\hat{Q}_t$, an unbiased sample of $Q^{\pi}(s_t, a_t)$. A truncated or bootstrapped return needs a separate bias analysis.

The cost of this fix is one index. It is the cheapest variance reduction in reinforcement learning, and in the numbers of the Act 1 widget it removes about half.

Fix 2 — a baseline: subtract anything action-independent

Subtract from each reward-to-go a quantity $b(s_t)$ that does not depend on the action taken:

$$\nabla_{\theta} J = \mathbb{E} \left[\sum_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (\hat{Q}_t - b(s_t)) \right]$$

THE BASELINE IS FREE

it adds *exactly* zero bias

$$\mathbb{E}_{a \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a | s) b(s)] = b(s) \int \frac{\nabla_{\theta} \pi_{\theta}(a | s)}{\pi_{\theta}(a | s)} \pi_{\theta}(a | s) da = b(s) \nabla_{\theta} \int \pi_{\theta}(a | s) da = b(s) \nabla_{\theta} 1 = 0$$

A useful baseline is $b(s) = V^{\pi}(s) = \mathbb{E}[G_t | s_t = s]$. It predicts the return typical of this state, so $G_t - b(s)$ asks whether the action did better than expected.

The value baseline is practical, but not always the exact minimum-variance baseline: the latter also weights by the squared score magnitude. A poorly chosen baseline can increase variance.

Treat the baseline as fixed in the actor gradient. Training it on independent or previously collected data avoids the extra dependence created by fitting and evaluating on the same action sample. The exact weighted formula is in Backup 3.

Temperature thread — make the better heater action more likely

shared teaching example · predict before revealing the calculation

For two actions, let $Q(\text{off}) = -6.7$ and $Q(\text{heat}) = -2.9$. The policy heats with $p = \sigma(\theta)$. At $\theta = 0$, $p = 0.5$ and $dp/d\theta = 0.25$. Treat the Q estimates as fixed for this actor step.

Predict: Which way should θ move? Does the fact that both action values are negative change the preferred direction?

Calculate and check. $J = (1 - p)(-6.7) + p(-2.9)$, so $dJ/d\theta = 0.25[-2.9 - (-6.7)] = \mathbf{0.95}$. Gradient ascent raises θ and makes heating more likely. The current value baseline is **-4.8**.

Relative action quality determines this update. A useful action can have a negative return when every outcome has a cost.

Try it — both action values shift down by 100

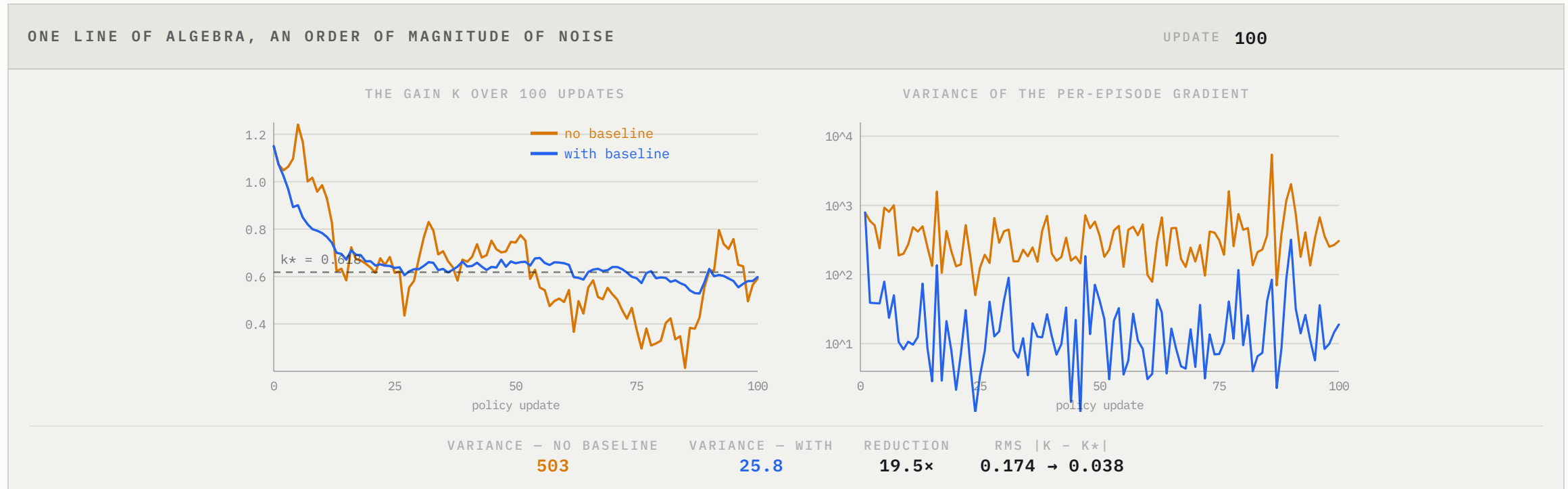
work independently · reveal only after writing an answer

Change the fixed Q estimates to -106.7 and -102.9 . Keep the same policy parameter. Recalculate the actor gradient and the value baseline.

Check your answer. The difference remains 3.8, so the gradient remains **0.95**. The baseline becomes **-104.8** . Subtracting that state-only baseline leaves the advantages unchanged.

A common offset changes the reported value but not this policy preference. Baseline invariance requires an action-independent subtraction.

The baseline, measured



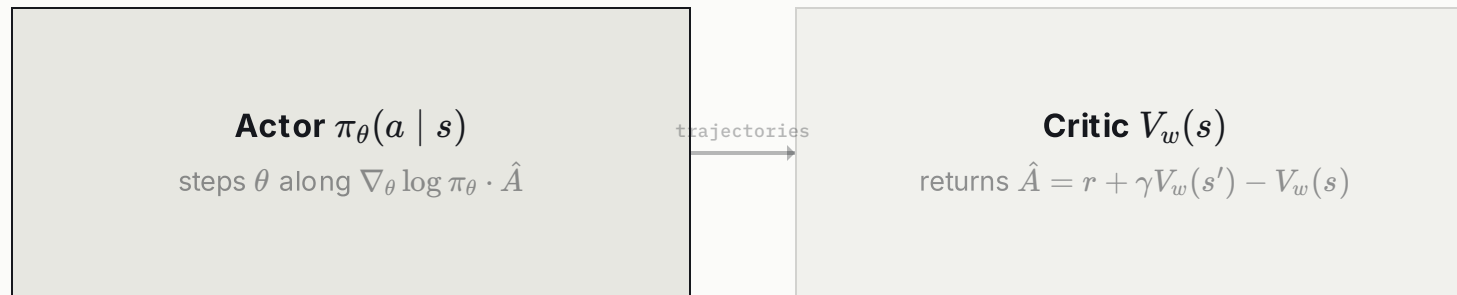
The same task, the same seed, the same starting gain, ten episodes per update — the only difference is whether $b(s_t)$ is subtracted. Left: the gain over 100 updates. Right: the variance of the per-episode gradient, on a log axis. The baseline changes **the mean of the estimator not at all** and its spread by nearly **twenty-fold**, which is the difference between a run that settles on k^* and one that keeps wandering past it.

Fix 3 — the advantage, and the actor–critic

Put the two together. With $b(s) = V^\pi(s)$ and $\hat{Q}_t \rightarrow Q^\pi(s_t, a_t)$, what multiplies the score is the **advantage**:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \quad \text{— “how much better than average was this action?”}$$

And now we need a *learned* V . Learn both:



That $\hat{A}$ is a **TD error** — Lecture 8's engine, running inside a policy update. An actor proposes actions and a critic estimates their consequences. Actor–critic methods combine policy optimization with value estimation; the course columns help explain the ingredients, not exclusive origins.

Baseline versus bootstrap — separate the two operations

Suppose the observed full return is **4.6**, the baseline is **2**, and the next-state critic is **3**. With reward 1 and $\gamma = 0.9$:

SIGNAL	CALCULATION	WHAT IT RELIES ON
MC minus baseline	$4.6 - 2 = 2.6$	the observed return
one-step TD advantage	$1 + 0.9(3) - 2 = 1.7$	a learned continuation value

A fixed action-independent baseline cancels in expectation. An inaccurate continuation value generally does not. **Actor-critic can trade some bias for faster, less noisy updates.**

At a true terminal state, continuation value is zero. GAE combines multiple TD errors to control this trade-off.

Two dials on the same idea

what the field does with the actor–critic once it has one

-
- **A3C / A2C** — run many actors in parallel on separate copies of the environment. Their rollouts provide more diverse recent experience without a large replay buffer. A2C synchronises updates; A3C permits parameter lag, so neither parallelism nor freshness means exact sample independence. (*Mnih et al., 2016*)
 - **n -step returns** — bootstrap after n steps rather than at the end: $R_t = \sum_{i=1}^n \gamma^{i-1} r_{t+i} + \gamma^n V(s_{t+n})$. A full Monte Carlo return continues to termination with zero bootstrap; one-step actor–critic uses $n = 1$.
 - **GAE** — do not pick n ; average all of them geometrically, $\hat{A}_t^{\text{GAE}(\gamma, \lambda)} = \sum_{l \geq 0} (\gamma \lambda)^l \delta_{t+l}^V$. (*Schulman et al., 2016*)
-

GAE is **TD(λ) for the advantage** — Lecture 8's bias–variance dial, moved from the value to the thing that trains the policy.

A3C and A2C — collect several short rollouts, then learn

original policy-RL PDF pp. 44–47 · parallel collection, not arbitrary replay

Each worker interacts with its own environment copy and forms an n -step target. For the main lecture's undiscounted episodic convention,

$$G_t^{(n)} = \sum_{j=0}^{n-1} r_{t+j+1} + V_w(s_{t+n}), \quad \hat{A}_t = G_t^{(n)} - V_w(s_t).$$

If an episode terminates within the window, stop the reward sum there and use zero continuation value. Discounted versions retain the corresponding powers of γ .

METHOD	COORDINATION
A3C	workers apply updates asynchronously; local parameters can lag behind the shared ones
A2C	collect a batch from workers, then update synchronously

Parallel workers diversify recent experience. This does **not** make samples independent or justify uncorrected use of arbitrarily old off-policy data.

Off-policy policy gradients — correct the distribution you sampled

original policy-RL PDF pp. 39–40 · why reusing data needs care

At a fixed state, data from behaviour policy β can estimate an expectation under π_θ using importance ratios:

$$\mathbb{E}_{a \sim \pi_\theta} [h(s, a)] = \mathbb{E}_{a \sim \beta} \left[\frac{\pi_\theta(a \mid s)}{\beta(a \mid s)} h(s, a) \right].$$

This requires $\beta(a \mid s) > 0$ wherever the target policy assigns probability. If $\pi(A \mid s) = 0.8$ but $\beta(A \mid s) = 0.4$, observed A actions receive weight **2**.

That ratio corrects the **action distribution at this state**. It does not by itself correct which states the behaviour policy visits or give an exact gradient of the original on-policy return. Off-policy actor–critic methods need their own objective and assumptions.

Check — what a baseline does

REINFORCE's gradient estimate is unbiased but very noisy. Subtracting a state-dependent baseline $b(s)$ from the return:

- ☐ A Reduces variance but introduces bias, which must be corrected later
- ☒ B Preserves the expected gradient; a well-chosen action-independent baseline can reduce variance
- ☐ C Reduces bias but leaves variance unchanged
- ☐ D Has no effect on either — it only rescales the learning rate

The score-function identity gives $\mathbb{E}[\nabla \log \pi \cdot b(s)] = 0$ for any b that does not depend on the action, so subtracting it is **free**. Choosing $b(s) = V(s)$ turns the return into the advantage — "was this action better than average here?" — which is the signal you actually wanted, and is what makes the actor–critic architecture worth its second network.

— ACT 3

Continuous control: DDPG

Q3. Lecture 8 handed over a wall. This is where it falls.

The wall Lecture 8 left



The source deck states the debt in one sentence: *"although DQN solves problems with high-dimensional observation spaces, it can only handle **discrete and low-dimensional action spaces**".* Consider two common choices for using a Q-network with continuous actions.

OPTION 1 – DISCRETISE THE ACTION

One output head per action, as in DQN. Take $-1 \leq a \leq 1$ and cut it into bins of width Δa .

What is a good Δa ? Coarse bins mean poor control resolution; fine bins mean n^d heads for d joints. And the optimum may lie **between grid points**.

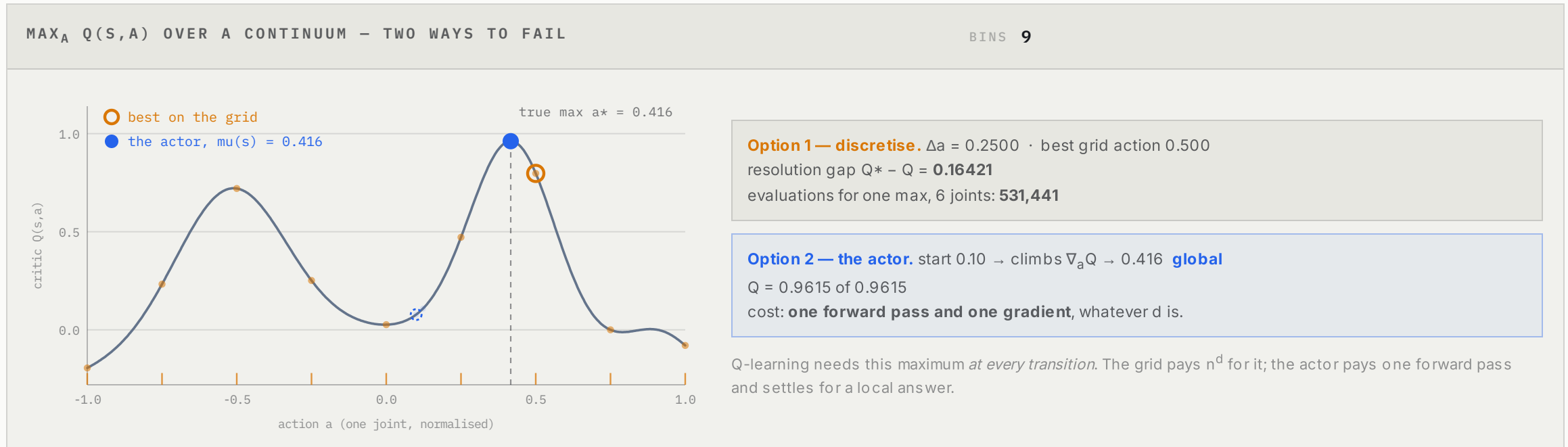
OPTION 2 – FEED THE ACTION IN

One network $Q_w(s, a)$ with the action as an input, so the action stays continuous.

But the update needs $\max_{a'} Q_w(s', a')$, and Q_w is **non-convex in a'** . That is a global optimisation, at every transition, forever.

Discretization changes the action set; generic continuous maximization adds an inner numerical search. Structured critics can make that search easier. Here we pursue a different practical choice: **learn an actor that proposes the action**, then improve it using the critic.

The wall, measured



One state, one continuous action, and the critic's $Q(s, \cdot)$ across it. Nine bins already cost $9^6 = 531,441$ evaluations per transition on a six-joint arm — and still miss the peak by 0.16. Twenty-one bins close the gap and cost **85,766,121**. The actor emits its action in **one forward pass** and refines it by following $\nabla_a Q$: cheap, and only ever *local* — move its start and watch it settle on the wrong hill, which exposes local optimization. TD3 addresses critic-related errors and SAC changes the objective; neither guarantees finding the best peak.

The deterministic fix — the actor climbs the critic

Make the actor **deterministic**, $a = \mu_\theta(s)$, and let it ascend the critic's action-gradient: (*Silver et al., 2014*)

$$\nabla_\theta J(\mu_\theta) = \mathbb{E}_{s \sim \rho^\mu} \left[\nabla_\theta \mu_\theta(s) \nabla_a Q(s, a) \Big|_{a=\mu_\theta(s)} \right]$$

This is how the intractable $\max_a Q$ is finally beaten: **the actor follows $\nabla_a Q$ uphill instead of searching for its summit.**

Determinism is not a stylistic preference. Compare the Bellman expectation in the two cases:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r, s' \sim E} \left[r + \gamma \mathbb{E}_{a' \sim \pi} [Q^\pi(s', a')] \right] \quad \text{versus} \quad Q^\mu(s_t, a_t) = \mathbb{E}_{r, s' \sim E} \left[r + \gamma Q^\mu(s', \mu(s')) \right]$$

With a deterministic μ the **inner expectation disappears** and only the environment is left inside the expectation. Two consequences, both decisive: the critic can use **off-policy transitions with adequate coverage**, and the actor need not sample an action integral. Arbitrary unsupported data and inaccurate critics do not provide the theorem's guarantee.

DDPG — deterministic policy gradient, plus every DQN stabiliser

Lillicrap et al., 2015 — DQN (2013) → DPG (2014) → DDPG (2016)

THE TWO LOSSES

Critic, by Bellman error on a replayed minibatch:

$$y_i = r_i + \gamma(1 - d_i) Q_{w^-}(s_i', \mu_{\theta^-}(s_i')),$$

$$L(w) = \frac{1}{N} \sum_i (Q_w(s_i, a_i) - y_i)^2.$$

Here s_i' is the next state stored with sample i , and d_i indicates true termination. **Actor**, by the deterministic policy gradient:

$$\nabla_{\theta} J \approx \frac{1}{N} \sum_i \nabla_a Q_w(s_i, a) \big|_{a=\mu_{\theta}(s_i)} \nabla_{\theta} \mu_{\theta}(s_i)$$

THE STABILISERS, RETURNING

- **replay buffer** — off-policy is now permitted, so use it;
- **target networks** w^-, θ^- , updated *softly* every step, $w^- \leftarrow \tau w + (1 - \tau)w^-$, rather than copied every C steps;
- both are Lecture 8's deadly-triad fixes, unchanged. The paper's own verdict on ablating them: "**target networks matter a lot**".

How the actor gradient appears in code. In code the actor loss is written `pi_loss = -Q(s, mu(s))` and `.backward()` is called; autograd's chain rule $\partial_{\theta} Q(s, \mu_{\theta}(s)) = \partial_a Q \cdot \partial_{\theta} \mu_{\theta}$ implements the actor's local chain-rule factor. The theorem also specifies the value function and state visitation distribution; replay-based DDPG uses approximations to them.

Follow a critic uphill — a scalar actor calculation

At one state let $Q(a) = 5 - (a - 2)^2$ and let the actor output $a = \theta$. Initially $\theta = 0$.

$$\frac{dQ}{da} = -2(a - 2), \quad \frac{da}{d\theta} = 1.$$

A step of size 0.1 gives $\theta_{\text{new}} = 0 + 0.1(4) = 0.4$. The critic's score rises from **1** to **2.44**.

The actor follows the **critic's slope**, without differentiating the environment. If the critic is wrong around 0.4, an improved predicted value may still mean worse real performance.

$\mu_\theta(s)$ and K — compare their feedback role

Put Lecture 9's Act 3 and this act side by side. Both produce feedback, using different assumptions and procedures.

	LQR (LECTURE 9)	DDPG (HERE)
the controller	$u = -Kx$	$a = \mu_\theta(s)$
how it is obtained	solve the Riccati equation	ascend $\nabla_a Q$ from samples
does it need the model?	yes — A, B, Q, R	no — sampled transitions
what it is	an $m \times n$ matrix	a neural network
optimality	global under LQR assumptions	no general guarantee of a local or global optimum

Same role — a state-to-action feedback rule — but different learning procedures and optimality guarantees.

The Act 1 widget uses a simple LQ benchmark with a known Riccati answer to assess a learned gain. Success on this example illustrates the feedback role; it is not a general convergence guarantee for policy gradients or DDPG.

A deterministic actor explores nothing

A deterministic policy has no randomness to explore with, and the obvious repair — act uniformly at random — is a poor one. The source deck gives three reasons, and the third is unusual in this course:

- random actions drive the policy update into regions where the critic **is not accurate** (Lecture 5's adversarial-optimiser hazard, in an RL costume);
- exploring *unseen states* needs action that is **consistent along the episode**, not resampled each step;
- and on real hardware, **rapid random input changes can strain actuators**.

THE PENDULUM ARGUMENT

the source deck's own picture

Swinging a pole upright takes a *sustained* push in one direction, then alternation near the top. Independent zero-mean noise often cancels across steps and may explore sustained maneuvers inefficiently, so a uniformly random policy might eventually find the swing-up — after enormously many samples.

Hence the **Ornstein-Uhlenbeck** process, $dx_t = -\kappa x_t dt + \sigma dW_t$, added to $\mu_\theta(s)$: noise that is *temporally correlated*, so exploration pushes rather than jitters.

TD3 — keep the deterministic actor, repair three critic-related failures

Fujimoto, van Hoof & Meger · ICML 2018 · Addressing Function Approximation Error in Actor-Critic Methods

A DDPG actor can exploit an erroneously high critic value. TD3 changes **the target and update timing**. Here d is the terminal indicator and ϵ is zero-mean Gaussian target noise.

CHANGE	OPERATION	PURPOSE
Clipped double Q	use the smaller of two target-critic values	reduce overly optimistic bootstrap targets
Delayed policy updates	update the actor less often than the critics	allow critic estimates to improve before the next actor step
Target policy smoothing	add clipped noise to the target action	average over nearby actions instead of rewarding a narrow spurious peak

$$\begin{aligned}\tilde{a}' &= \text{clip}(\mu_{\theta^-}(s') + \text{clip}(\epsilon, -c, c), a_{\min}, a_{\max}), \\ y &= r + \gamma(1 - d) \min_{i=1,2} Q_{w_i^-}(s', \tilde{a}').\end{aligned}$$

The two critics each regress to this target. On delayed steps the actor climbs the **first** critic. Target smoothing noise is separate from exploration noise used to collect data. [TD3](#)

Calculate a TD3 target — and keep its guarantee in proportion

Let $r = 1$, $\gamma = 0.9$, and $d = 0$. A target actor proposes 0.5; the clipped target noise is 0.1, so the smoothed action is **0.6** within the bounds $[-1, 1]$.

VALUE AT THAT SAME NEXT ACTION	ESTIMATE
First target critic	$Q_{w_1^-}(s', 0.6) = 8$
Second target critic	$Q_{w_2^-}(s', 0.6) = 6$

$$y_{\text{TD3}} = 1 + 0.9 \min(8, 6) = 6.4.$$

Using the first estimate alone would give **8.2** at this action. If the transition were terminal, both targets would instead be **1**. An illustrative delay of two means two critic updates for each actor update.

A smaller target addresses one error mechanism; it is not a certified lower bound on the true return. Critics can share errors. Lecture 12 keeps these TD3 mechanics and adds a behaviour-cloning term for fixed-data learning.

SAC — reward good actions while preserving policy entropy

Haarnoja et al. · ICML 2018 · an entropy-regularized objective

SAC trains a **stochastic actor** from replay. Its discounted episodic objective includes policy entropy:

$$J(\pi) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{T-1} \gamma^t (r_{t+1} + \alpha_H \mathcal{H}(\pi(\cdot \mid s_t))) \right].$$

Here $\alpha_H > 0$ is the **entropy temperature**, not the gradient step size. Consider a one-step, two-action illustration with rewards 2 and 1.8, and $\alpha_H = 0.5$:

POLICY	EXPECTED REWARD	ENTROPY, USING NATURAL LOGS	REWARD + TEMPERATURE × ENTROPY
Always choose the first action	2	0	2
Choose each with probability 0.5	1.9	$\log 2 \approx 0.693$	2.247

The mixture wins **between these two candidates** because the objective values diversity as well as reward. This discrete illustration explains the entropy term; continuous SAC uses a probability density and differential entropy. [Soft Actor-Critic](#)

SAC in the learning loop — a soft target and a stochastic actor

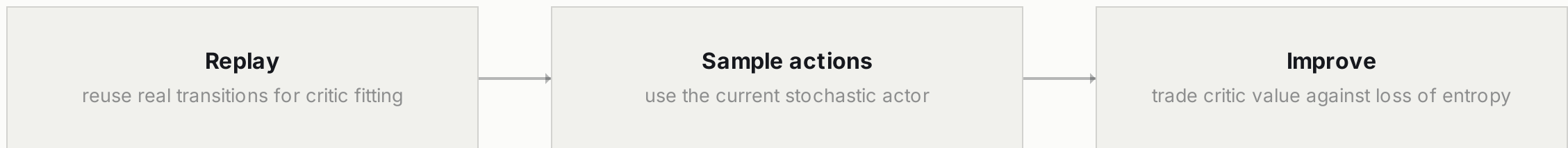
twin-Q formulation from Soft Actor-Critic Algorithms and Applications · 2018/2019

Draw a next action $a' \sim \pi_\theta(\cdot \mid s')$ for each replayed transition. Train both critics toward

$$y = r + \gamma(1 - d) \left[\min_{i=1,2} Q_{w_i^-}(s', a') - \alpha_H \log \pi_\theta(a' \mid s') \right].$$

The actor minimizes, over replayed states and newly sampled current-policy actions,

$$L_\pi = \mathbb{E}_{s \sim D, a \sim \pi_\theta} \left[\alpha_H \log \pi_\theta(a \mid s) - \min_i Q_{w_i}(s, a) \right].$$



The actor can use the reparameterization idea from Lecture 6. The original ICML version used a separate value network; these equations are the later twin-Q variant. Neither replay nor entropy fixes missing coverage in a fixed offline log. [Algorithms and Applications](#)

Check — the deterministic gradient

DDPG trains a critic $Q(s, a)$ and a deterministic actor $\mu(s)$. How does the actor get its gradient?

- ☐ A By finite differences on the environment
- ☐ B By sampling actions and weighting them by their returns, as in REINFORCE
- ☒ C By backpropagating through the critic: $\nabla_{\theta} Q(s, \mu_{\theta}(s))$ — the critic is differentiable, so the actor climbs it
- ☐ D By solving the $\arg \max$ exactly at each step and regressing onto the result

The critic is a differentiable surrogate for "how good is this action here", so the actor can be moved uphill on it by the chain rule — no action sampling inside this deterministic actor derivative; the state samples still come from data. Notice what has just been re-created: an optimiser climbing a **learned model of the objective**, which is Lecture 5's setup exactly. Lecture 12 shows it failing in the same way.

— ACT 4

Stepping without falling

Q4. We have a gradient and we have tamed its variance. How far along it dare we step?

The step-size trap unique to RL



In supervised learning a bad step costs one bad update; the next minibatch arrives from the same fixed dataset and pulls you back. In RL it is worse, because

the policy generates its own next batch of data.

A step that pushes π_θ too far lands in a region where the policy is bad — and now *every* subsequent sample comes from that bad policy. The gradient estimated there points somewhere else again. The damage is **self-reinforcing**, and there is no fixed dataset to fall back on.

This is also why the parameter step size is the wrong thing to control. A tiny change in θ can produce a large change in *behaviour* (a sensitive neural network or a Gaussian policy with very small variance), and a large one can produce none at all. What must be bounded is the distance between the old and new **action distributions**.

A trust region, in policy space

Maximise an importance-weighted advantage, subject to staying inside a **KL ball** around the policy that collected the data: (*Schulman et al., 2015*)

$$\max_{\theta} \mathbb{E} \left[\frac{\pi_{\theta}(a | s)}{\pi_{\theta_{\text{old}}}(a | s)} \hat{A} \right] \quad \text{s.t.} \quad \mathbb{E} \left[D_{\text{KL}} \left(\pi_{\theta_{\text{old}}}(\cdot | s) \parallel \pi_{\theta}(\cdot | s) \right) \right] \leq \delta$$

The ratio reweights old-policy actions. A local linear objective and quadratic KL give the natural-gradient step (Backup 5). Practical average-KL TRPO does **not** by itself guarantee monotonic real return; the theoretical bound uses stronger assumptions.

THIS IS LECTURE 1'S TRUST REGION, MOVED

the `trust-region` widget of Chapter 1, in policy space

LECTURE 1 • SUCCESSIVE CONVEXIFICATION	TRPO
a quadratic model of f	a linear model of the advantage
trust radius ρ in parameter space	KL radius δ in behaviour space
accept if the model was believable, else shrink	backtrack until estimated KL and surrogate improvement pass their checks

Shared idea: **restrict an update to where its local model is useful**. The metrics and acceptance tests differ.

PPO — discourage large changes with a clipped objective

Practical TRPO approximates a KL-constrained step. PPO uses a simpler surrogate with a similar intent: with $r(\theta) = \pi_\theta(a \mid s) / \pi_{\theta_{\text{old}}}(a \mid s)$, (Schulman et al., 2017)

$$J^{\text{CLIP}}(\theta) = \mathbb{E} \left[\min \left(r(\theta) \hat{A}, \text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A} \right) \right]$$

No second-order machinery, no conjugate gradient, no line search. When the ratio leaves $[1 - \epsilon, 1 + \epsilon]$ *in the direction the update was pushing it*, the objective goes flat and **the incentive to push further is simply removed**.

With shared actor-critic parameters the practical objective adds two terms — $-c_1(V_\theta(s) - V_{\text{target}})^2$ for the critic and $+c_2 H(\pi_\theta(\cdot \mid s))$, an entropy bonus for exploration. The entropy term is the seed of maximum-entropy RL and of SAC.

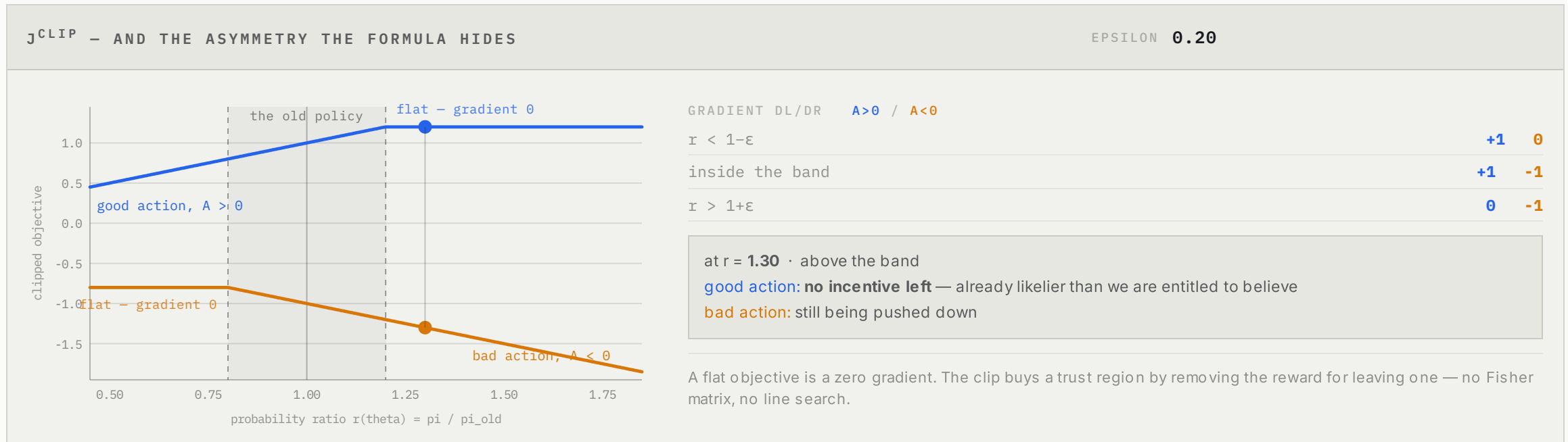
Calculate the clip — the sign of advantage matters

Let $\epsilon = 0.2$. For one sampled action, compare $r\hat{A}$ with $\text{clip}(r, 0.8, 1.2)\hat{A}$, and keep the **smaller** number.

ADVANTAGE	RATIO	ORDINARY TERM	CLIPPED TERM	PPO TERM
+2	1.4	2.8	2.4	2.4
-2	0.6	-1.2	-1.6	-1.6
-2	1.4	-2.8	-2.4	-2.8

PPO removes the incentive to increase a good action too much or decrease a bad one too much. It still penalizes making a bad action more likely. **Clipping is not a hard bound on every probability ratio or on KL.**

The clip, and the asymmetry nobody mentions



The clipped objective as a function of the ratio, for a good action ($\hat{A} > 0$) and a bad one ($\hat{A} < 0$). Read the slopes: for $\hat{A} > 0$ the gradient is **exactly zero above $1 + \epsilon$** — no reward for making a good action still likelier. For $\hat{A} < 0$ it is zero *below* $1 - \epsilon$ but **stays alive above $1 + \epsilon$** : an action already too probable and known to be bad keeps being pushed down. The clip only removes the incentive that would take you out of the region.

PPO and SAC — compare complete learning procedures

Schulman et al. (2017) · Haarnoja et al. (2018)

QUESTION	PPO	SAC
Which data?	recent rollouts under the collection policy; reused for a limited number of epochs	a replay buffer containing transitions from earlier policies
What guides the actor?	an estimated advantage with a clipped probability-ratio objective	soft Q estimates and an entropy term
What controls the update?	clipping changes incentives for probability-ratio changes	actor/critic optimization and an entropy temperature
How is exploration represented?	stochastic collection policy; an entropy bonus is often added	entropy is part of the stated maximum-entropy objective

Both can use stochastic neural policies in continuous control. PPO clipping is not a hard KL guarantee; SAC's replay is not a guarantee that arbitrary offline data will suffice.

Compare **return, environment steps and computation** under the same evaluation protocol. TD3 is a deterministic off-policy comparison; Lecture 12 removes further interaction. [PPO](#) · [SAC](#)

Check — why large policy changes are discouraged

PPO discourages some large policy updates through its clipped surrogate. Why are large changes risky?

- ☐ A The gradient estimate becomes biased
- ☐ B The value function stops converging
- ☐ C The policy becomes deterministic too quickly
- ☒ D The data was collected under the old policy, so a large step moves into a region the batch says nothing about — and the estimate of the improvement stops being valid

On-policy data is only evidence about policies near the one that gathered it. A large update leaves that neighbourhood, and the objective being maximised is then a surrogate evaluated well outside its region of validity — Lecture 1's trust region, arrived at from a completely different direction, and the same adversarial-optimiser problem as Lecture 5.

— CLOSING

Closing

Both lineages have now had the model taken away. The grid is full.

Where we are — both lineages, now data-driven

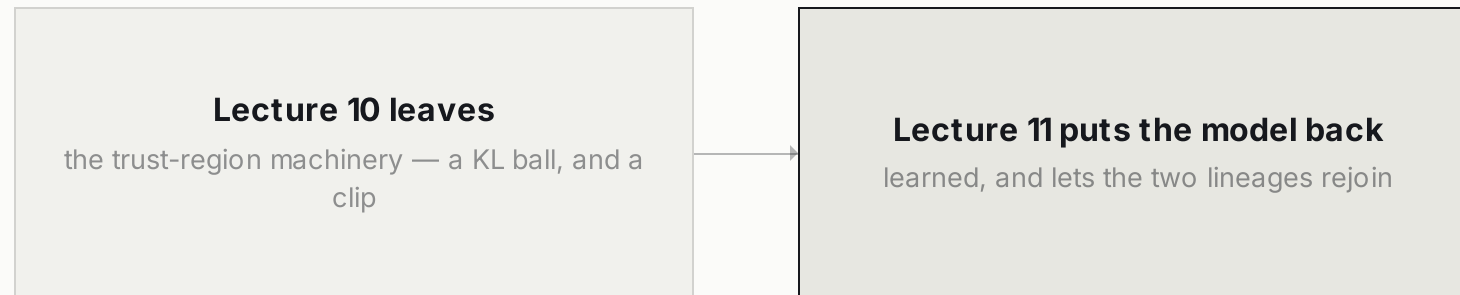
	LINEAGE A • OR / DYNAMIC PROGRAMMING	LINEAGE B • CONTROL THEORY
MODEL-BASED (ORIGIN)	MDP & Dynamic Programming LECTURE 7	Optimal Control / Planning LECTURE 9
DATA-DRIVEN (EXTENSION)	Value-Based RL LECTURE 8	Policy-Based RL LECTURE 10

Two traditions, each with a model-based origin and a data-driven extension, and the same move relates each pair:

	MODEL-BASED ORIGIN	WHAT IS DELETED	DATA-DRIVEN EXTENSION
Lineage A • OR	MDP & DP (<i>Lec 7</i>)	the model (P, R)	value-based RL (<i>Lec 8</i>)
Lineage B • Control	optimal control (<i>Lec 9</i>)	the dynamics f	policy-based RL (<i>Lec 10</i>)

One orphaning move, made twice: **delete the model, sample instead.**

What we hand on



Lecture 1 and TRPO both control where a local approximation is trusted, using different metrics and acceptance rules. PPO clipping instead changes incentives in its sampled objective; it does not impose a hard KL bound. Lecture 12 adds another issue: evaluating a changed policy using a fixed log.

Lecture 11 adds a learned model: MBPO uses SAC to learn from short synthetic rollouts, while Dreamer trains an actor and critic in latent imagination. Lecture 12 later starts from TD3 and adds behavior cloning when fresh interaction is unavailable.

Policy gradients learn feedback from experience;
their guarantees differ from model-based optimal
control.

The dynamics never solved, only experienced — and the controller learned, not derived.

Questions?

Lecture 8 learns action values; this lecture learns feedback policies. Explain the policy-gradient estimate first, then distinguish **DDPG/TD3's deterministic actor**, **PPO's clipped update**, and **SAC's entropy-regularized learning**. Their objectives, data and estimators differ. Which learned model could help each one use fewer real interactions?

— APPENDIX

Backup slides

Complete arguments, kept out of the narrative.

Reading guide — learn feedback, then test the critic and the data regime

one main idea to explain, one comparison, one application

ROLE	READ OR REVISIT	QUESTION TO ANSWER
Core	Schulman et al., <i>Proximal Policy Optimization Algorithms</i> (2017)	How is an advantage estimate used while discouraging large policy changes?
Compare	Haarnoja et al., <i>Soft Actor-Critic</i> (ICML 2018) ; Fujimoto et al., <i>TD3</i> (ICML 2018)	What changes when learning from replay, using twin critics or rewarding entropy?
Apply	The scalar feedback benchmark and the TD3/entropy calculations in this lecture	Does a better critic score imply a better real controller?

First derive REINFORCE and DDPG. Then compare PPO and SAC as two complete learning procedures. TD3 supplies the prerequisite for Lecture 12's TD3+BC; A3C, GAE and exact trust-region derivations provide supporting context.

Backup 1—the policy gradient theorem, the DP route (i): a recursion

Use a fixed start distribution μ , bounded rewards, and $\gamma < 1$: $J(\theta) = \sum_s \mu(s) V^\pi(s)$. Define $\phi(s) = \sum_a \nabla_\theta \pi_\theta(a | s) Q^\pi(s, a)$.

Differentiate $V^\pi(s) = \sum_a \pi_\theta(a | s) Q^\pi(s, a)$ and use the Bellman equation:

$$\nabla_\theta V^\pi(s) = \phi(s) + \gamma \sum_{s'} P^\pi(s' | s) \nabla_\theta V^\pi(s').$$

The transition law has no direct parameter derivative, but the **future policy** still does. Repeated substitution produces discounted state visitations. This is different from differentiating an objective with an arbitrary stationary start distribution that itself depends on θ .

Backup 2 — the policy gradient theorem, the DP route (ii): unrolled

Let $d_\gamma^\pi(s) = (1 - \gamma) \sum_{t \geq 0} \gamma^t P_\pi(S_t = s \mid S_0 \sim \mu)$. This is a normalized discounted visitation distribution.

$$\nabla_\theta J(\theta) = \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d_\gamma^\pi, a \sim \pi_\theta} [\nabla_\theta \log \pi_\theta(a \mid s) Q^\pi(s, a)].$$

The factor $1/(1 - \gamma)$ can be absorbed into a learning rate; the **state weighting cannot simply be discarded**. Finite-horizon episodic versions use a sum over time instead.

The trajectory route and the Bellman route agree when they describe the same objective and the same visitation weights.

Backup 3 — the baseline: unbiased, and the best one

At a fixed state, put $z = \nabla_{\theta} \log \pi_{\theta}(a \mid s)$. Since $\mathbb{E}[z \mid s] = 0$, subtracting a fixed $b(s)$ leaves the mean unchanged.

For the variance of **one score-weighted contribution**, minimize $\mathbb{E}[\|z\|^2 (G - b)^2 \mid s]$. Differentiation gives

$$b^*(s) = \frac{\mathbb{E}[\|z\|^2 G \mid s]}{\mathbb{E}[\|z\|^2 \mid s]}.$$

This equals $V^{\pi}(s) = \mathbb{E}[G \mid s]$ when the score weight is constant or suitably uncorrelated with return. Correlations across time add further considerations for a whole-trajectory variance optimum.

A value critic is a useful baseline, not a universal exact variance minimizer. Keep its output fixed during the actor derivative; same-sample fitting can introduce statistical dependence.

Backup 4 — GAE, and the λ dial

With $\delta_t^V = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$, the k -step advantage estimators telescope:

$$\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V = -V(s_t) + r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^k V(s_{t+k})$$

$k = 1$ is the one-step TD advantage: low variance, biased by whatever error V carries. $k = \infty$ is the Monte-Carlo advantage: unbiased, and as noisy as REINFORCE. GAE takes the exponentially weighted average of all of them, (*Schulman et al., 2016*)

$$\hat{A}_t^{\text{GAE}(\gamma, \lambda)} := (1 - \lambda) (\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \cdots) = \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{t+l}^V$$

which collapses to a single running sum of TD errors. $\lambda = 0$ recovers one-step actor-critic, $\lambda = 1$ recovers Monte-Carlo.

This is **TD(λ) applied to the advantage** rather than to the value, and the discount γ is doing double duty: as the problem's discount, and as a variance-reduction parameter that downweights delayed effects at the cost of bias. Lecture 8's Backup 1 drew the same dial for MC-versus-TD; it is the same dial.

Backup 5 — TRPO's natural-gradient step, in closed form

Maximise $L(\theta) = \mathbb{E}[r(\theta)\hat{A}]$ subject to $\bar{D}_{\text{KL}}(\theta_{\text{old}}, \theta) \leq \delta$.

Step 1 — local models. To first and second order about θ_{old} ,

$$L(\theta) \approx g^\top (\theta - \theta_{\text{old}}), \quad \bar{D}_{\text{KL}} \approx \frac{1}{2} (\theta - \theta_{\text{old}})^\top H (\theta - \theta_{\text{old}})$$

with $g = \nabla_\theta L$ and H the Fisher information matrix. The KL has zero gradient at θ_{old} , which is why its leading term is quadratic.

Step 2 — a linear objective on an ellipsoid. The maximiser lies along the natural-gradient direction $H^{-1}g$, scaled to the KL radius:

$$\theta_{\text{new}} = \theta_{\text{old}} + \sqrt{\frac{2\delta}{g^\top H^{-1}g}} H^{-1}g$$

Step 3 — practice. $H^{-1}g$ is obtained by conjugate gradient without ever forming H , and the step is backtracked until the sample-estimated KL and surrogate improvement pass the line-search checks — which is Lecture 1's accept-or-shrink test, verbatim.

The derivation assumes an invertible positive-definite local Fisher matrix and a nonzero gradient. Implementations use damping and estimated quantities. PPO removes the second-order solve, but does not inherit a hard trust-region or monotonic-improvement guarantee.

Backup 6 — the policy-based zoo, placed

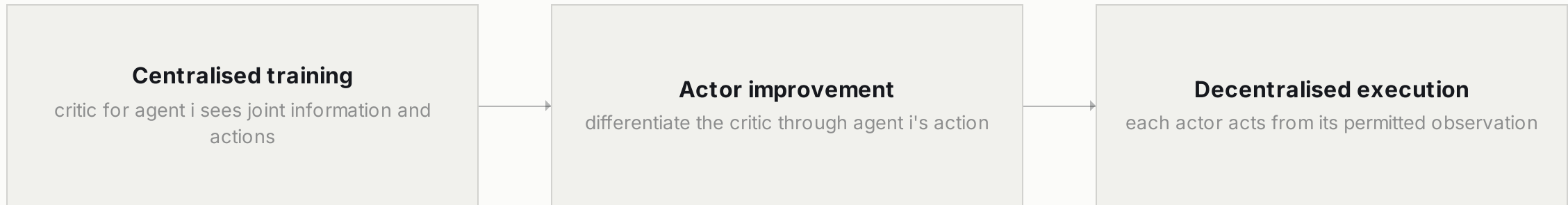
These methods share policy optimization, but differ in objectives, estimators, data distributions and update rules. SAC explicitly includes entropy; PPO uses a clipped surrogate. A single vanilla policy-gradient formula does not specify every algorithm below.

METHOD	KEY IDEA	REFERENCE
REINFORCE	Monte-Carlo policy gradient, no critic	Williams, 1992
A3C / A2C	parallel actor-critic; recent diverse rollouts	Mnih et al., 2016
GAE	λ -weighted advantage	Schulman et al., 2016
DPG / DDPG	deterministic actor climbs $\nabla_a Q$; off-policy	Silver 2014; Lillicrap 2015
TD3	twin critics, delayed actor — the overestimation fix	Fujimoto et al., 2018
TRPO	approximate KL trust-region update	Schulman et al., 2015
PPO	clipped surrogate; the workhorse	Schulman et al., 2017
SAC	maximum-entropy off-policy actor-critic	Haarnoja et al., 2018

Source extension — MADDPG changes the number of decision makers

original policy-RL PDF pp. 71–74 · preview for IE579

With several agents, a critic can use the joint state and actions during training, while each actor uses its own observation when deployed.



This is **centralised training with decentralised execution**. Other agents' changing policies make the learning problem different from single-agent DDPG; actor gradients do not by themselves guarantee convergence to a Nash equilibrium.

This extension belongs to the multi-agent face of the course map. It is a preview, not a prerequisite for Lecture 11.

The translation table — optimal control, learned

Every object of Lecture 9 has a data-driven counterpart. Keep this in sight all lecture.

LECTURE 9 (OPTIMAL CONTROL)	LECTURE 10 (POLICY-BASED RL)	WHAT REPLACES THE MODEL
value field $V(t, x)$, the HJB equation	critic V_w, Q_w	a <i>learned</i> value
feedback law $u = \gamma(t, x), u = -Kx$	actor $\pi_\theta(a \mid s) / \mu_\theta(s)$	a <i>learned</i> controller
known dynamics $\dot{x} = f(x, u)$	sampled transitions (s, a, r, s')	the environment itself
Pontryagin's costate, trajectory optimisation	REINFORCE, trajectory sampling	the score-function trick
solve Riccati / HJB	gradient ascent on $J(\theta)$	stochastic optimisation
controllability of (A, B)	sufficient exploration	a reachability assumption, either way

Read column 3. Nothing here is a new *principle* — it is Lecture 9's programme with the one piece we no longer own quietly swapped out. **Policy gradient is optimal control with f deleted and replaced by data.** DDPG's actor $\mu_\theta(s)$ is the learned cousin of LQR's gain K ; REINFORCE also uses trajectories, but its score-function gradient is a different mathematical argument from Pontryagin's costate equations.

One honest detour — why not identify the system?

The most literal idea, and the exact rhyme of the detour Lecture 8 took: estimate the dynamics from data, then run Lecture 9 unchanged.

$$(\hat{A}, \hat{B}) = \arg \min_{A, B} \sum_t \|x_{t+1} - Ax_t - Bu_t\|^2 \implies \text{solve the Riccati equation for } \hat{K}$$

This is **system identification**, it is the oldest idea in adaptive control, and it is perfectly valid.

But notice the same waste Lecture 8 noticed. To act we only ever need the controller — $m \times n$ numbers, or one network — yet here we first estimate a *whole dynamics model*, and then we act as though it were true. Why estimate f at all,

when we could estimate **the controller directly?**

That question — *skip the model, learn the thing that acts* — is the entire policy-based programme, and it is Lecture 8's question with Q^* replaced by π_θ . (The model comes back, deliberately and learned, in Lecture 11.)