Applied Computational Mathematics Lab

TEDDY LAZEBNIK'S COURSE · LECTURE 10

Reinforcement Learning

Lecture 10 · Policy gradients and actor-critics: learning the policy directly: stochastic policies, the score function trick, the policy gradient theorem, actor-critics, trust regions and continuous actions



Teddy Lazebnik · Applied Computational Mathematics Laboratory

BEFORE WE START

Why not learn the policy directly?

- So far we took the long way round: learn values, then read a policy off them; or learn a model, plan, and get a policy.
- Even "policy π is better than π'" was defined by values.
- And small errors in values can flip the greedy policy to a completely different action.
- If we care about optimal behaviour, why not learn a policy directly?

VLADIMIR VAPNIK, 1998

"Do not solve a more general problem as an intermediate step."



IN PLAIN WORDS

A well-known joke in statistics: what's the best way to beat a casino? Don't play. The dynamics of poker or roulette are anything but simple; the optimal policy is trivial.

BEFORE WE START

Today's map

PART 1

Policy-based RL

Why learn policies, and why stochastic ones; an aliased gridworld.

PART 2

The score function trick

Objectives, gradient ascent, REINFORCE, baselines and softmax.

PART 3

The policy gradient theorem

From one step to whole episodes; a short corridor with switched actions.

PART 4

Actor-critics

Critics cut the noise; trust regions keep steps safe.

PART 5

Continuous actions

Gaussian policies, deterministic policy gradients and Cacla.

READING

Sutton & Barto (2018)

Chapter 13. Williams (1992); Sutton et al. (2000); Silver et al. (2014).



IN PLAIN WORDS

By the end you should be able to derive the score-function gradient, run REINFORCE and one-step actor-critic by hand, and explain what a baseline and a trust region do.

Applied Computational Mathematics Lab

PART 1

Policy-based RL

Learn the thing you actually want.



Three roads to a good policy

	PROS	CONS
Model-based	"easy" to learn a model (supervised); learns all there is to know from the data	spends compute and capacity on irrelevant details; planning is non-trivial and expensive
Value-based	easy to generate a policy ($\pi(a \mid s) = \mathbb{I}(a = \operatorname{argmax}_a q(s, a))$); close to the true objective; well understood	still not the true objective; small value errors can cause larger policy errors
Policy-based	the right objective!	see the next slides

- All of these generalise in different ways. Sometimes a model is easier to learn (simple dynamics); sometimes a policy is ("always move forward" is optimal).



IN PLAIN WORDS

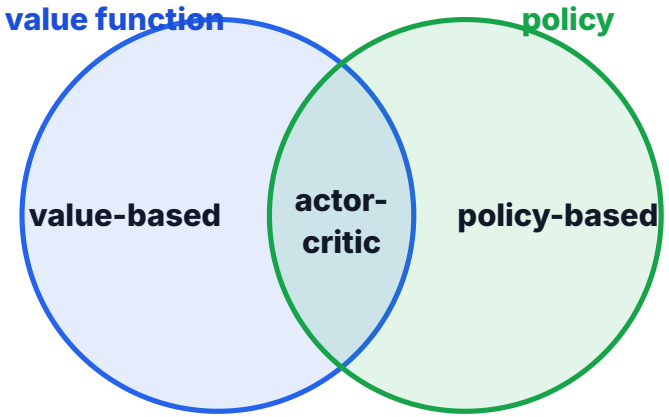
Complex world, trivial policy: then learning a model or every value is a lot of work for nothing.

Parametrising the policy

- Before, we approximated values, $v_{\mathbf{w}}(s) \approx v_\pi(s)$ and $q_{\mathbf{w}}(s,a) \approx q_\pi(s,a)$, and generated a policy from them (e.g. greedy).
- Now we parametrise the policy directly:

$$\pi_\theta(a \mid s) = p(a \mid s, \theta)$$

- This lecture is about model-free RL.
- **Value-based**: learn values; the policy is implicit (e.g. ε-greedy). **Policy-based**: no values; learn the policy. **Actor-critic**: learn both.





IN PLAIN WORDS

A deterministic policy gives probabilities of 0 or 1; a stochastic one anything in between. The same notation covers both.

Advantages and disadvantages

ADVANTAGES

policy-based RL

The true objective. Extends easily to high-dimensional or continuous actions. Can learn stochastic policies. Sometimes the policy is simple while values and models are complex.

DISADVANTAGES

policy-based RL

Can get stuck in local optima. The knowledge learned can be specific and not generalise well. On its own, it doesn't extract all useful information from the data.



IN PLAIN WORDS

Sound familiar? Gradient descent with backpropagation has the same problems: they come from optimising over functions, not from RL.

TRY IT

A robot that must cross a slippery floor: modelling the friction is hard, but the best policy may just be "walk slowly, straight ahead".

Why stochastic policies?

- In an MDP there is always an optimal deterministic policy.
- But most problems are not fully observable, especially with function approximation, where states blur together. Then the optimal policy may be **stochastic**.
- The search space is smoother for stochastic policies, so we can use gradients.
- Stochastic policies also explore during learning.

EXAMPLE

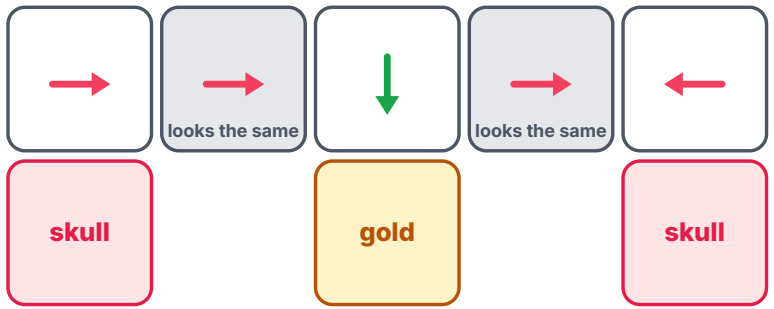
Rock-paper-scissors against a clever opponent: any deterministic choice gets exploited. Playing each with probability 1/3 cannot be.



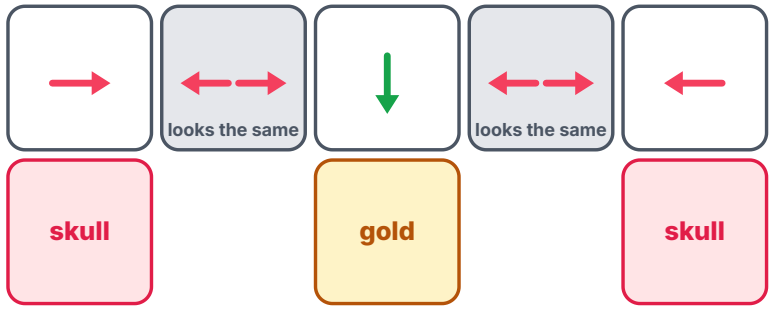
IN PLAIN WORDS

Board games are fully observed, maybe. Almost everything else isn't.

Example: an aliased gridworld



Deterministic: “go right” in both grey cells. From the right half it bounces between the last two cells forever.



Stochastic: left or right with probability 1/2 in the grey cells. Our drawing, after the lecture’s example.

- The agent only senses the walls around it: $\phi(s) = (1, 0, 1, 0)$ (walls up and down) in **both** grey cells.
- Our runs, random start: the deterministic policy reaches the gold 60% of the time; the rest it walks back and forth until stopped.
- The coin-flip policy reached the gold every time, in 3.8 steps on average (exactly 3.8).

TRY IT

Chance of going left ten times in a row: $0.5^{10} \approx 0.0010$. Left forever: 0.



IN PLAIN WORDS

Learning the policy directly, the probabilities can be anything, not just 50/50: this is not random tie-breaking.

Check yourself: Part 1

CHECK YOURSELF

1. Name one advantage of value-based RL over policy-based RL, and one of policy-based over value-based.

CHECK YOURSELF

2. Why can the best policy be stochastic when the state is only partly observed?

CHECK YOURSELF

3. In the aliased gridworld, what is the probability the coin-flip policy goes right five times in a row from a grey cell?





HINT

Answers are at the end of the deck.

Applied Computational Mathematics Lab

PART 2

The score function trick

One identity makes the policy gradient something we can sample.



What do we maximise?

- Episodic: the expected return from the start, i.e. the value of the start state, with start distribution d_0:

$$J_G(\theta) = \mathbb{E}_{S_0 \sim d_0, \pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1}\right] = \mathbb{E}_{S_0 \sim d_0}\left[v_{\pi_\theta}(S_0)\right]$$

- Continuing: the average reward per step, with $d_\pi(s)$ the long-run fraction of time spent in s:

$$J_R(\theta) = \mathbb{E}_{\pi_\theta}[R_{t+1}] = \sum_s d_{\pi_\theta}(s) \sum_a \pi_\theta(a \mid s) \sum_r p(r \mid s, a)\, r$$

FROM CLASS

A diving policy: every minute, go one metre deeper, no matter what. At some point that's a return of minus infinity. Judge a policy by where it actually takes you.



IN PLAIN WORDS

In chess, d_0 is a single board. In blackjack, it's every starting pair of cards. The objective only sees states the start distribution can reach.

Policy optimisation

- Policy-based RL is an optimisation problem: find θ maximising $J(\theta)$.
- We focus on **stochastic gradient ascent**: often efficient, and easy to use with deep networks.

$$\Delta\theta = \alpha \nabla_\theta J(\theta), \qquad \nabla_\theta J(\theta) = \left(\frac{\partial J(\theta)}{\partial \theta_1}, \ldots, \frac{\partial J(\theta)}{\partial \theta_n} \right)^\top$$

- Gradient-free alternatives exist: hill climbing, simulated annealing, genetic algorithms, evolution strategies. Sometimes more robust; usually more expensive.
- Stochastic policies help make $J(\theta)$ smooth (typically): nudge a probability a little, and the expected return changes a little.



IN PLAIN WORDS

We ask for a differentiable policy, almost everywhere: everywhere except a set of size zero. A gradient is a promise about the neighbourhood; if the function jumps around, the promise is worthless.

The policy gradient for a bandit

- A one-step case (a contextual bandit): $J(\theta) = \mathbb{E}_{\pi_\theta}[R(S, A)]$, over states d (not depending on π, for now) and actions π.
- We can't sample R_{t+1} and take its gradient: it's just a number, and doesn't depend on θ. Instead we use:

$$\nabla_\theta \, \mathbb{E}_{\pi_\theta}[R(S, A)] = \mathbb{E}_{\pi_\theta}\left[R(S, A) \, \nabla_\theta \log \pi_\theta(A \mid S)\right]$$

- The right-hand side is an expected gradient we **can** sample. The stochastic update:

$$\theta_{t+1} = \theta_t + \alpha \, R_{t+1} \, \nabla_\theta \log \pi_{\theta_t}(A_t \mid S_t)$$



IN PLAIN WORDS

Increase the probability of actions with high rewards. In expectation this follows the true gradient: a pure, unbiased stochastic gradient algorithm.

NAME

REINFORCE (Williams, 1992). We met it as the gradient bandit in Lecture 2; now we see where it comes from.

The score function trick

- Let $r_{sa} = \mathbb{E}[R(S,A) \mid S=s, A=a]$. Then

$$\begin{aligned}\nabla_\theta \mathbb{E}_{\pi_\theta}[R(S,A)] &= \nabla_\theta \sum_s d(s) \sum_a \pi_\theta(a \mid s)\, r_{sa} = \sum_s d(s) \sum_a r_{sa}\, \nabla_\theta \pi_\theta(a \mid s) \\ &= \sum_s d(s) \sum_a r_{sa}\, \pi_\theta(a \mid s) \frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} = \sum_s d(s) \sum_a \pi_\theta(a \mid s)\, r_{sa}\, \nabla_\theta \log \pi_\theta(a \mid s) \\ &= \mathbb{E}_{d,\pi_\theta}\left[R(S,A)\, \nabla_\theta \log \pi_\theta(A \mid S)\right]\end{aligned}$$



IN PLAIN WORDS

Multiply and divide by π: the gradient of π over π is the gradient of $\log \pi$. What's left is an expectation again.

A TRICK WORTH STEALING

It shows up all over machine learning, whenever you need a gradient through a random choice.

Checking the trick by sampling

SAMPLES	ARM 1	ARM 2	ARM 3
10	+0.254	-0.015	-0.239
100	-0.047	+0.257	-0.210
1,000	-0.100	+0.354	-0.255
100,000	-0.097	+0.345	-0.248
exact	**-0.098**	**+0.344**	**-0.246**

Our run: a 3-armed softmax bandit with preferences (0, 0.5, −0.5), so $\pi = (0.307, 0.506, 0.186)$; mean rewards (1, 2, 0) plus noise. Estimates of the gradient with respect to the three preferences.

- With 10 samples the estimate is a mess: two of three components point the wrong way.
- With 100, in the neighbourhood; with 1,000, close; with 100,000, off by a few thousandths.



IN PLAIN WORDS

Unbiased, but noisy. That's the story of this whole lecture: right on average, all over the place on any single try.

Baselines reduce variance

- For any b that does not depend on the action (it may depend on the state):

$$\mathbb{E}\big[b\,\nabla_\theta \log \pi(A_t \mid S_t)\big] = \mathbb{E}\Big[\sum_a \pi(a \mid S_t)\, b\,\nabla_\theta \log \pi(a \mid S_t)\Big] = \mathbb{E}\Big[b\,\nabla_\theta \sum_a \pi(a \mid S_t)\Big] = \mathbb{E}[b\,\nabla_\theta 1] = 0$$

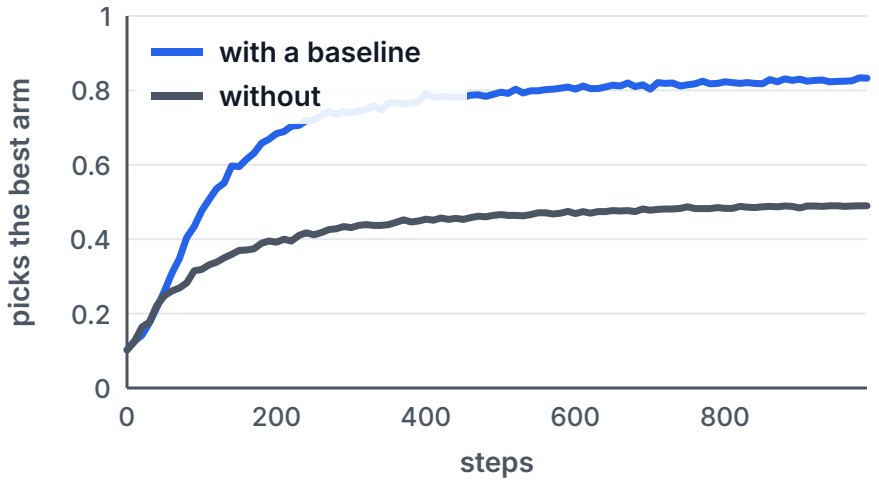
- So we can subtract a baseline without changing the expected gradient:

$$\theta_{t+1} = \theta_t + \alpha\big(R_{t+1} - b(S_t)\big)\nabla_\theta \log \pi_{\theta_t}(A_t \mid S_t)$$



IN PLAIN WORDS

When every action looks great, the baseline asks the right question: better than usual, or worse?



Our runs: a 10-armed bandit with rewards around +4, $\alpha = 0.1$, 2,000 runs. Over the last 100 steps: 83% with the running-average baseline, 49% without.

Example: a softmax policy

- Action preferences $h(s, a)$; probability proportional to their exponentials:

$$\pi_\theta(a \mid s) = \frac{e^{h(s,a)}}{\sum_b e^{h(s,b)}}, \qquad \nabla_\theta \log \pi_\theta(A_t \mid S_t) = \underbrace{\nabla_\theta h(S_t, A_t)}_{\text{chosen preference}} - \underbrace{\sum_a \pi_\theta(a \mid S_t) \nabla_\theta h(S_t, a)}_{\text{expected preference}}$$

- The chosen action goes up; every action shares the cost; scaled by how good the reward was.
- If every reward is positive, even a bad chosen action goes up: the good ones just go up faster.



IN PLAIN WORDS

Scores of 8 and 9, or 800 and 900, rank the actions the same. Only the relative scores matter, but the noise is very different.

TRY IT

Two actions, preferences $(1, 0)$: $\pi = (0.731, 0.269)$. You pick action 1; $R - b = 2$; $\alpha = 0.1$.

$\Delta h_1 = 0.1 \times 2 \times (1 - 0.731) = +0.054.$

$\Delta h_2 = 0.1 \times 2 \times (0 - 0.269) = -0.054.$

Check yourself: Part 2

CHECK YOURSELF

4. Why can't we just take the gradient of the sampled reward R_{t+1} with respect to θ?

CHECK YOURSELF

5. Why does subtracting a baseline $b(s)$ not change the expected gradient?

CHECK YOURSELF

6. Softmax with two actions, preferences $(0, 0)$. You pick action 2, $R - b = -1$, $\alpha = 0.5$. New preferences?



Applied Computational Mathematics Lab

PART 3

The policy gradient theorem

From one step to whole episodes.



The policy gradient theorem (episodic)

- The approach extends to multi-step MDPs: replace the reward with the long-term return G_t, or the value $q_\pi(s, a)$.
- There are two theorems (Sutton et al., 2000): for the average return per episode, and for the average reward per step.

THEOREM

For any differentiable policy π_θ and start distribution d_0, the gradient of $J(\theta) = \mathbb{E}[G_0 \mid S_0 \sim d_0]$ is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t \, q_{\pi_\theta}(S_t, A_t) \, \nabla_\theta \log \pi_\theta(A_t \mid S_t) \,\middle|\, S_0 \sim d_0\right]$$

where $q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$.



IN PLAIN WORDS

Policy gradients don't need to know the dynamics. Surprising: shouldn't we know how the policy influences the states?

Proof, part 1: trajectories

- Take a trajectory $\tau = S_0, A_0, R_1, S_1, A_1, R_2, S_2, \ldots$ with return $G(\tau)$. The score function trick works for whole trajectories:

$$\nabla_\theta J_\theta(\pi) = \nabla_\theta \, \mathbb{E}[G(\tau)] = \mathbb{E}\big[G(\tau) \, \nabla_\theta \log p(\tau)\big]$$

$$\begin{aligned}
\nabla_\theta \log p(\tau) &= \nabla_\theta \log \big[p(S_0)\, \pi(A_0 \mid S_0)\, p(S_1 \mid S_0, A_0)\, \pi(A_1 \mid S_1) \cdots\big] \\
&= \nabla_\theta \big[\log p(S_0) + \log \pi(A_0 \mid S_0) + \log p(S_1 \mid S_0, A_0) + \log \pi(A_1 \mid S_1) + \cdots\big] \\
&= \nabla_\theta \big[\log \pi(A_0 \mid S_0) + \log \pi(A_1 \mid S_1) + \cdots\big]
\end{aligned}$$

- So $\nabla_\theta J_\theta(\pi) = \mathbb{E}_\pi\big[G(\tau) \, \nabla_\theta \sum_{t=0}^{T} \log \pi(A_t \mid S_t)\big]$.

TRY IT

A two-step trajectory: $p(S_0) = 1$, $\pi(A_0 \mid S_0) = 0.5$, $p(S_1 \mid S_0, A_0) = 0.8$, $\pi(A_1 \mid S_1) = 0.3$. Then $\log p(\tau) = \log 1 + \log 0.5 + \log 0.8 + \log 0.3$. Only $\log 0.5$ and $\log 0.3$ depend on θ.



IN PLAIN WORDS

The log of a product is a sum of logs. The dynamics terms don't depend on θ, so their gradients vanish. Products are painful; sums are easy.

FROM CLASS

The reward is not part of the policy. Dynamics are physics; we want the ball in the basket. Physics doesn't care what we want.

Proof, part 2: only the future counts

$$\nabla_\theta J_\theta(\pi) = \mathbb{E}_\pi\Big[\sum_{t=0}^{T} G(\tau)\,\nabla_\theta \log \pi(A_t \mid S_t)\Big] = \mathbb{E}_\pi\Big[\sum_{t=0}^{T}\Big(\sum_{k=0}^{T}\gamma^k R_{k+1}\Big)\nabla_\theta \log \pi(A_t \mid S_t)\Big]$$

$$= \mathbb{E}_\pi\Big[\sum_{t=0}^{T}\Big(\sum_{k=t}^{T}\gamma^k R_{k+1}\Big)\nabla_\theta \log \pi(A_t \mid S_t)\Big] \quad \text{(earlier rewards: zero in expectation)}$$

$$= \mathbb{E}_\pi\Big[\sum_{t=0}^{T}\gamma^t G_t\,\nabla_\theta \log \pi(A_t \mid S_t)\Big] = \mathbb{E}_\pi\Big[\sum_{t=0}^{T}\gamma^t q_\pi(S_t, A_t)\,\nabla_\theta \log \pi(A_t \mid S_t)\Big]$$

- An action at time t can't change rewards that came before it (the baseline argument again), so each step needs only the return from t on: G_t.
- And $\mathbb{E}[G_t \mid S_t, A_t] = q_\pi(S_t, A_t)$. That's the theorem.



IN PLAIN WORDS

We use the real G_t: we played to the end, so we know it. Why guess when you know the answer?

The episodic algorithm

- We can sample $\nabla_\theta J$ from a whole episode. Typically we split it into per-step updates:

$$\Delta\theta_t = \gamma^t G_t \nabla_\theta \log \pi_\theta(A_t \mid S_t), \qquad \mathbb{E}_\pi\Big[\sum_t \Delta\theta_t\Big] = \nabla_\theta J_\theta(\pi)$$

- In practice people often drop the γ^t factor: $\Delta\theta_t = G_t \nabla_\theta \log \pi_\theta(A_t \mid S_t)$.
- That's okay-ish: as if, at every step, the episode could have started right there. Or view it as a slightly biased gradient.

TRY IT

Corridor rewards $-1, -1, -1$, $\gamma = 1$: $G_0 = -3$, $G_1 = -2$, $G_2 = -1$. Every action is pushed down, the early ones most. In expectation shorter episodes are pushed down less, so the policy still drifts the right way; a baseline (Part 4) makes it much less noisy.

REINFORCE (EPISODIC)

1. Play an episode with π_θ.
2. For each step, compute the return G_t.
3. For each step: $\theta \leftarrow \theta + \alpha\, G_t\, \nabla_\theta \log \pi_\theta(A_t \mid S_t)$.
4. Repeat.



IN PLAIN WORDS

No model, no reward function: only samples, and a policy you can differentiate.

The average-reward theorem

THEOREM (AVERAGE REWARD)

For any differentiable policy, the gradient of $J(\theta) = \mathbb{E}[R \mid \pi]$ is

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[q_{\pi_\theta}(S_t, A_t) \nabla_\theta \log \pi_\theta(A_t \mid S_t) \right]$$

with the differential value $q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} - \rho + q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$ and $\rho = \mathbb{E}_\pi[R_{t+1}]$, a global average. Equivalently:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[R_{t+1} \sum_{n=0}^{\infty} \nabla_\theta \log \pi_\theta(A_{t-n} \mid S_{t-n}) \right]$$



IN PLAIN WORDS

Only the distance from the average reward matters: what hurts is the spread around it.

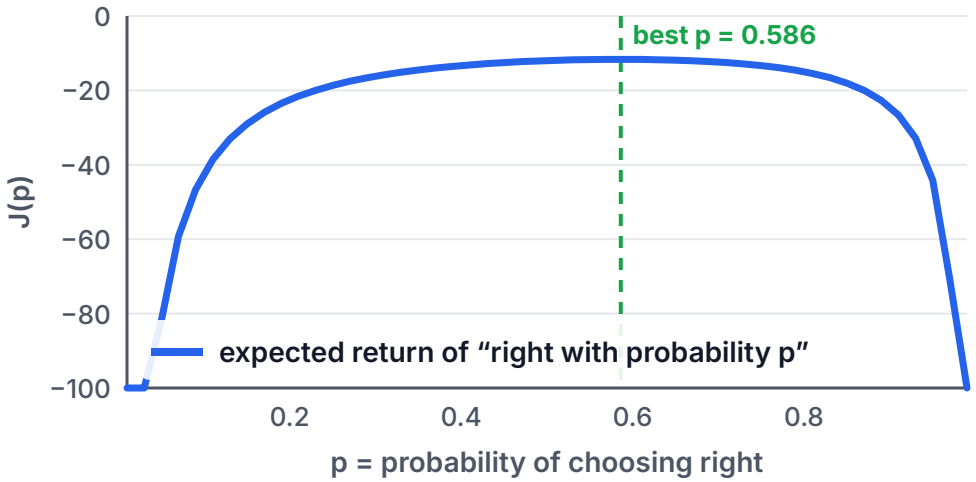
READ IT AS

The second form credits each reward to every earlier action: an eligibility trace for the policy.

Example: a short corridor with switched actions



Example 13.1 of Sutton & Barto (2018). Our drawing.



- All three cells look the same, so the policy is one number: the probability p of choosing right.
- Our exact computation: mostly right ($p = 0.95$) takes 44.2 steps on average; mostly left ($p = 0.05$), 82.1.
- The best policy goes right 59% of the time and needs about 11.66 steps. Stochastic again: no deterministic policy comes close.

TRY IT

REINFORCE from a poor start ($p \approx 0.19$), 100 runs of 1,000 episodes: 23.2 steps per episode over the first 50 episodes, 13.4 over the last 100. It never knew the middle cell was switched.

Check yourself: Part 3

CHECK YOURSELF

7. Why do the dynamics terms disappear from $\nabla_\theta \log p(\tau)$?

CHECK YOURSELF

8. Why can we replace the full return $G(\tau)$ by G_t in the term for step t?

CHECK YOURSELF

9. In the short corridor, why can't a deterministic policy reach the goal reliably?



Applied Computational Mathematics Lab

PART 4

Actor-critics

A critic calms the actor’s noise.



A critic as a baseline

- Recall $\mathbb{E}_\pi[b(S_t)\nabla \log \pi(A_t \mid S_t)] = 0$. A common baseline is $v_\pi(S_t)$:

$$\nabla_\theta J_\theta(\pi) = \mathbb{E}\Big[\sum_t \gamma^t \big(q_\pi(S_t, A_t) - v_\pi(S_t)\big)\nabla_\theta \log \pi(A_t \mid S_t)\Big]$$

- Estimate $v_{\mathbf{w}}(s) \approx v_\pi(s)$ explicitly, and sample $q_\pi(S_t, A_t) \approx G_t$.
- Reduce variance further by bootstrapping: $G_t \approx R_{t+1} + \gamma v_{\mathbf{w}}(S_{t+1})$. Less variance, a little bias.



IN PLAIN WORDS

$q - v$ measures what my specific action contributed, compared with how good the state already was. A plain-vanilla action gets a tiny gradient.

WHY NOISE HURTS

Two levers. One pays \$100 now and then and \$0 otherwise; the other pays \$1 every time. Three pulls each, and your estimates jump around; a jump in the estimates is a jump in the policy.

TRY IT

Arm A pays \$10 every time. Arm B costs \$10, but once in a thousand pulls pays \$1,000,000. In expectation B wins by far; with \$100 in your pocket, you pick A. Lower variance is sometimes a choice, and for learning it is almost always worth it.

Critics and the actor-critic algorithm

- A **critic** is a value function, learned by policy evaluation: what is v_{π_θ} for the current θ? Monte Carlo, TD, n-step TD: nothing new.
- **Critic**: update $\mathbf{w}$ of $v_{\mathbf{w}}$ by TD or MC. **Actor**: update θ by the policy gradient.

ONE-STEP ACTOR-CRITIC

1. Initialise s, θ, $\mathbf{w}$.
2. Sample $A_t \sim \pi_\theta(S_t)$; observe R_{t+1}, S_{t+1}.
3. $\delta_t = R_{t+1} + \gamma v_{\mathbf{w}}(S_{t+1}) - v_{\mathbf{w}}(S_t)$ (TD error, an estimate of the advantage).
4. $\mathbf{w} \leftarrow \mathbf{w} + \beta\, \delta_t \nabla_{\mathbf{w}} v_{\mathbf{w}}(S_t)$ (TD(0)).
5. $\theta \leftarrow \theta + \alpha\, \delta_t \nabla_\theta \log \pi_\theta(A_t \mid S_t)$ (policy gradient, ignoring γ^t).



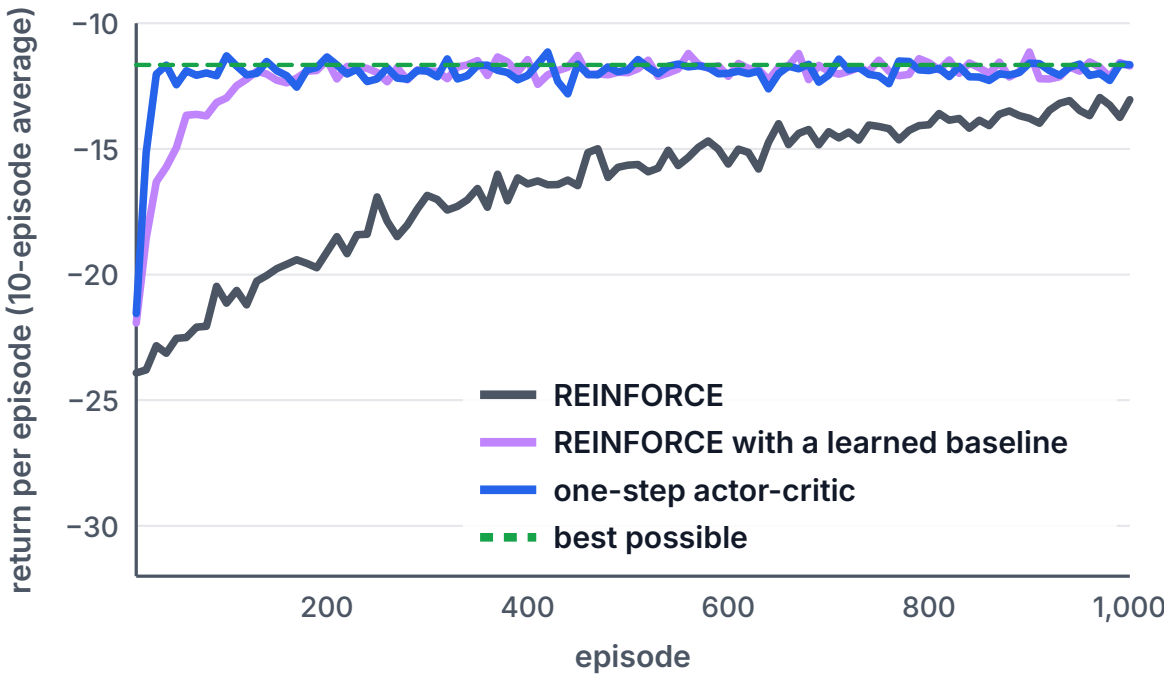
IN PLAIN WORDS

Everything moves at once, a bit like SARSA. If one part moves too fast, the other chases a moving target, so usually the critic learns a little faster than the actor.

TRY IT

$v(s) = 2$, $v(s') = 3$, reward 1, $\gamma = 0.9$: $\delta = 1 + 2.7 - 2 = 1.7$. The action was better than expected: the critic raises $v(s)$, and the actor makes that action more likely.

The corridor, three ways



Our runs: 100 runs of 1,000 episodes, the same poor start for every method.

METHOD	FIRST 50	LAST 100
REINFORCE	-23.2	-13.4
with a baseline	-17.5	-11.9
actor-critic	-14.6	-11.9



IN PLAIN WORDS

The critic helps most early on, while the policy is still bad.

A FAIR WARNING

Each method has its own step sizes, not tuned to the bone. The ordering is the point, not the decimals.

Trust regions

- Take care: bad policies lead to bad data. Unlike supervised learning, learning and data are not independent.
- One way to increase stability: limit how much the policy changes between updates, for example with the Kullback–Leibler divergence:

$$\mathrm{KL}(\pi_{\text{old}} \,\|\, \pi_\theta) = \mathbb{E}_S\Big[\int \pi_{\text{old}}(a \mid S) \log \frac{\pi_{\text{old}}(a \mid S)}{\pi_\theta(a \mid S)}\, da\Big]$$

- Then maximise $J(\theta) - \eta\,\mathrm{KL}(\pi_{\text{old}} \| \pi_\theta)$ for some η: the idea behind TRPO (Schulman et al., 2015), PPO and MPO.

STEP SIZE	RUNS STUCK ON A WORSE ARM	RUNS ON THE BEST ARM
0.2	0.0%	99.4%
1.0	0.3%	99.7%
5.0	22.9%	76.9%

Our runs: a 3-armed softmax bandit (success rates 0.2, 0.5, 0.8), 2,000 runs of 500 steps. “Stuck/best”: probability above 0.95. With step 0.05 no run had committed yet (best arm chosen 89% of the time on average).



IN PLAIN WORDS

Big steps learn fast when they’re right, and badly when they’re wrong. A trust region is a seat belt.

Check yourself: Part 4

CHECK YOURSELF

10. $v(s) = 5$, $v(s') = 4$, reward 0, $\gamma = 1$. What is the TD error, and does the actor make the action more or less likely?

CHECK YOURSELF

11. Why should the critic usually learn faster than the actor?

CHECK YOURSELF

12. Why does a bad policy hurt more in RL than a bad model does in supervised learning?



Applied Computational Mathematics Lab

PART 5

Continuous actions

Infinitely many actions: let a distribution do the exploring.



Continuous actions

- Pure value-based RL is non-trivial to extend to continuous actions: how to approximate $q(s, a)$? How to compute $\max_a q(s, a)$ at every step?
- Updating the policy parameters directly, continuous actions are easier. Most algorithms today work for discrete and continuous actions.
- Exploration in high-dimensional continuous spaces can still be hard.

EXAMPLE

An autonomous car picks a speed anywhere between 0 and 100. Discretise in steps of 1: 101 actions. Add steering and braking, and the count explodes.



IN PLAIN WORDS

Instead of a value for every speed, learn where to aim, and how much to wobble around it. Infinite vectors are no fun; vectors of size two are much better.

A Gaussian policy

- Mean $\mu_\theta(s)$, a function of the state; fixed variance σ^2 for simplicity (it can be learned too): $A_t \sim \mathcal{N}(\mu_\theta(S_t), \sigma^2)$. (Here μ_θ is the mean, not a behaviour policy.)

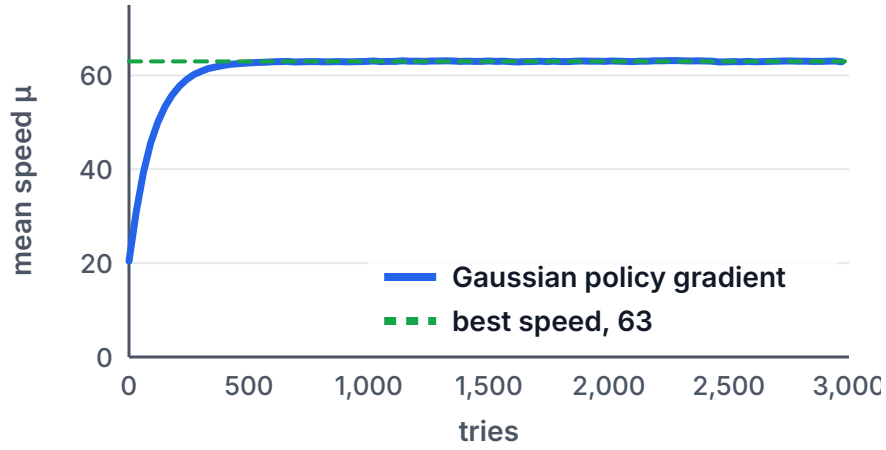
$$\nabla_\theta \log \pi_\theta(a \mid s) = \frac{a - \mu_\theta(s)}{\sigma^2} \nabla_\theta \mu_\theta(s), \qquad \theta_{t+1} = \theta_t + \beta\big(G_t - v(S_t)\big) \frac{A_t - \mu_\theta(S_t)}{\sigma^2} \nabla_\theta \mu_\theta(S_t)$$

- Intuition: if the return was high, move $\mu_\theta(S_t)$ toward the action you tried; if low, away from it.
- The exploration comes for free, from the spread σ.



IN PLAIN WORDS

The policy isn't Gaussian; it's a function. The distribution it returns over actions can be.



Our toy: pick a speed; reward $-((a-63)/10)^2$; start $\mu = 20$, $\sigma = 10$; 200 runs. Within 2 of 63 after about 309 tries.

Gradient ascent on the value

- Policy gradients work well, but don't strongly exploit the critic. If values generalise well, perhaps we can rely on them more:
- 1. estimate $q_{\mathbf{w}} \approx q_\pi$, e.g. with SARSA;
- 2. define a **deterministic** actor: $A_t = \pi_\theta(S_t)$;
- 3. improve the actor by gradient ascent on the value:

$$\Delta\theta \propto \frac{\partial q_\pi(s, \pi_\theta(s))}{\partial \theta} = \frac{\partial q_\pi(s,a)}{\partial a}\Big|_{a=\pi_\theta(s)} \frac{\partial \pi_\theta(s)}{\partial \theta}$$

- Known under various names: action-dependent heuristic DP (Werbos, 1990; Prokhorov & Wunsch, 1997), gradient ascent on the value (van Hasselt & Wiering, 2007), and today mostly the **deterministic policy gradient** (DPG; Silver et al., 2014). A form of policy iteration.



IN PLAIN WORDS

Move the action up the slope of q: in the direction of better actions, as the critic sees them.

TRY IT

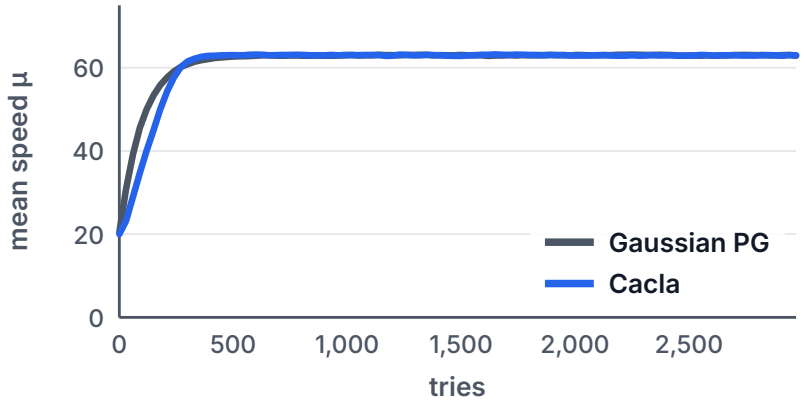
$q(s,a) = -(a-3)^2$ for one state; actor $\pi_\theta(s) = \theta = 1$. $\partial q/\partial a = -2(1-3) = 4$, $\partial \pi/\partial \theta = 1$: with step 0.1, θ moves to 1.4, toward 3.

Cacla

CONTINUOUS ACTOR-CRITIC LEARNING AUTOMATON

1. $a_t = \text{Actor}_\theta(S_t)$: the current action proposal.
2. Explore: $A_t \sim \pi(\cdot \mid S_t, a_t)$, e.g. $\mathcal{N}(a_t, \Sigma)$.
3. $\delta_t = R_{t+1} + \gamma v_{\mathbf{w}}(S_{t+1}) - v_{\mathbf{w}}(S_t)$.
4. Update $v_{\mathbf{w}}(S_t)$, e.g. by TD.
5. If $\delta_t > 0$: $\theta \leftarrow \theta + \beta(A_t - a_t)\nabla_\theta \text{Actor}_\theta(S_t)$.
6. If $\delta_t \leq 0$: don't update the actor.

- The error is defined in action space, not parameter space. The update size doesn't depend on the size of the values, and it never pushes away from bad actions.



Our toy again. Cacla is within 2 of 63 after about 287 tries. Multiply every reward by 100 (same step size): the Gaussian policy gradient blew up in all 200 runs; Cacla behaved exactly the same.



IN PLAIN WORDS

Sacrifice your queen, and a few moves later, checkmate. The sacrifice doesn't teach the actor "never do that"; the win afterwards does.

Check yourself: Part 5

CHECK YOURSELF

13. Gaussian policy, $\mu = 10$, $\sigma = 2$, you tried $a = 12$. What is $\partial \log \pi / \partial \mu$? If the advantage was positive, which way does μ move?

CHECK YOURSELF

14. Why does Cacla not care if all rewards are multiplied by 100?

CHECK YOURSELF

15. Why is $\max_a q(s, a)$ a problem for continuous actions?



WRAP-UP

Summary

- Learn the policy directly: the true objective, stochastic policies, continuous actions; but local optima and noisy gradients.
- The **score function trick**: $\nabla \mathbb{E}[R] = \mathbb{E}[R \nabla \log \pi]$. **REINFORCE** samples it.
- The **policy gradient theorem**: weight $\nabla \log \pi$ by q_π (or G_t); no model needed.
- **Baselines** and **critics** cut the variance without changing the expected gradient; **actor-critic** uses the TD error.
- **Trust regions** keep steps small; **Gaussian policies**, **DPG** and **Cacla** handle continuous actions.

$$\nabla_\theta J = \mathbb{E}\big[q_\pi(S, A)\, \nabla_\theta \log \pi_\theta(A \mid S)\big]$$

$$\theta \leftarrow \theta + \alpha\, \delta_t \nabla_\theta \log \pi_\theta(A_t \mid S_t)$$

ANSWERS

Answers to check yourself

- **1.** Value-based: easy to generate a policy, well understood. Policy-based: the true objective (or: continuous actions, stochastic policies).
- **2.** Different states look the same, and one deterministic action can't suit them all.
- **3.** $0.5^5 = 1/32$.
- **4.** The reward is a number from the environment; θ only changes which actions are chosen.
- **5.** $\sum_a \pi \nabla \log \pi = \nabla \sum_a \pi = \nabla 1 = 0$.
- **6.** $\pi = (0.5, 0.5)$: $\Delta h_2 = 0.5 \times (-1) \times 0.5 = -0.25$, $\Delta h_1 = +0.25$: $(0.25, -0.25)$.
- **7.** They don't depend on θ.
- **8.** Rewards before step t can't be affected by A_t; their terms average to zero.
- **9.** One action must suit all three look-alike cells, and the middle cell reverses it.
- **10.** $\delta = 0 + 4 - 5 = -1$: less likely.
- **11.** The actor's update relies on the critic's estimate; a stale critic gives a misleading direction.
- **12.** The policy collects its own data: a bad policy gathers unhelpful data.
- **13.** $(12 - 10)/4 = 0.5$: μ moves up, toward 12.
- **14.** It only uses the sign of δ and the action difference $A_t - a_t$.
- **15.** Infinitely many actions: the max is itself an optimisation at every step.

CREDITS

Sources and credits

- The order of topics follows *Lecture 9: Policy Gradients and Actor Critics* of the DeepMind × UCL Reinforcement Learning Lecture Series (2021), by Hado van Hasselt. This deck is not an official DeepMind or UCL material.
- The text, figures, examples and simulations were written, drawn and run for ACML. Numbers labelled “our run” come from our own code; examples taken from books or papers are credited where they appear.
- Sutton & Barto (2018), chapter 13: the short corridor (13.1), re-run by us. Williams (1992), REINFORCE; Sutton et al. (2000), the policy gradient theorems; Silver et al. (2014), DPG; van Hasselt & Wiering (2007), Cacla; Schulman et al. (2015), TRPO. The aliased gridworld follows the lecture’s example; the bandits and the speed toy are our own.
- Companion videos: **ACML Explains**, season 1, episodes 38–42. More from the lab: acml.teddylazebnik.com



Applied Computational Mathematics Lab

END OF LECTURE 10

Next: learning about many things

Lecture 11 returns to deep RL: general value functions, auxiliary tasks, distributional RL and other ways to learn more from the same data.

