Applied Computational Mathematics Lab

TEDDY LAZEBNIK'S COURSE · LECTURE 12

# Reinforcement Learning

Lecture 12 · Learning from people: imitation learning, behaviour cloning and DAgger, inverse RL and maximum entropy, learning from comparisons, RLHF, and its limits



Teddy Lazebnik · Applied Computational Mathematics Laboratory

BEFORE WE START

# Warm-up: true or false?

**STATEMENT (A)**

(a) Behaviour cloning needs to know the dynamics model.

**STATEMENT (B)**

(b) DAgger uses the expert's demonstrations, and after that needs no more help from the expert.

**STATEMENT (C)**

(c) Keeping each policy update inside a trust region (Lecture 10) guards against a single large, bad step.





**HOW TO USE THIS**

Guess now. Statements (a) and (b) are settled in Part 1; all three answers are at the end of the deck.

BEFORE WE START

# Why learn from people?

- Every lecture so far assumed someone hands the agent a reward.
- Try writing one for **driving well**: stay in the lane, don’t hit anything, don’t brake too hard, be polite at roundabouts… Every rule you add, an agent will find a loophole in.
- But a decent driver can just **show** you. One drive says more than ten pages of rules.
- And people can say which of two drives they prefer, even when they can’t say why.

**WHERE THIS MATTERS**

Driving, tying a surgical knot, folding a shirt, writing a helpful answer. Ask an expert for a formula and you get a shrug; ask them to do it, or to compare two attempts, and you get data.



**IN PLAIN WORDS**

Today the humans write the number. Or rather, they whisper it: one demonstration, or one comparison, at a time.

BEFORE WE START

# Today's map

PART 1

## Imitation learning

Behaviour cloning, compounding errors, and DAgger.

PART 2

## Inverse RL

Recover the reward; feature matching; why the answer isn't unique.

PART 3

## Maximum entropy IRL

Pick the least committed explanation; match visit counts.

PART 4

## Learning from comparisons

Kinds of human feedback; the Bradley–Terry model; winners and cycles.

PART 5

## RLHF

Reward models from comparisons, RL with a KL price, from backflips to chatbots.

PART 6

## Limits

Reward hacking, noisy labels, who labels, groups that disagree, simpler baselines.



IN PLAIN WORDS

By the end you should be able to explain why behaviour cloning drifts, derive the max-entropy IRL gradient, fit a Bradley–Terry model, and say what the KL term in RLHF protects against.

Applied Computational Mathematics Lab

PART 1

# Imitation learning

When showing is easier than explaining.

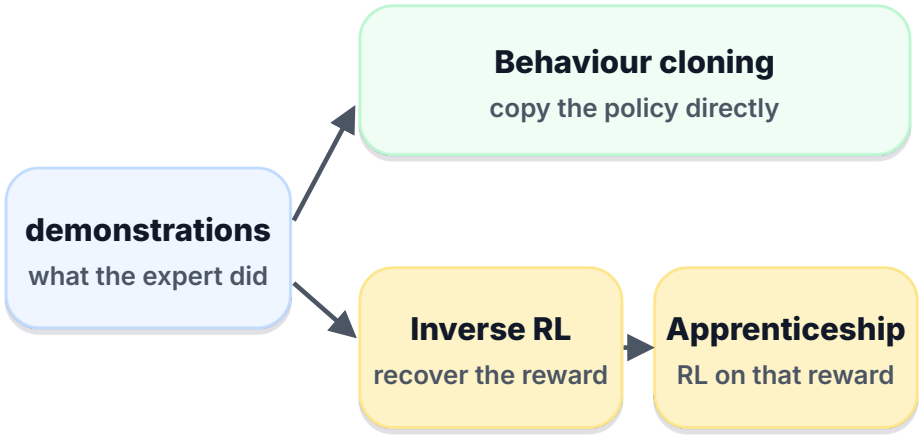


# Learning from demonstrations

- An expert provides a set of **demonstration trajectories**: sequences of states and actions.
- Imitation learning is useful when it is easier for the expert to demonstrate the behaviour than:
  - to specify a reward that would generate it, or
  - to specify the policy directly.

**PROBLEM SETUP**

Input: a state space and an action space; sometimes the transition model $P(s' \mid s, a)$; **no reward function**; and one or more demonstrations $(s_0, a_0, s_1, a_1, \ldots)$ with actions drawn from the expert's policy $\pi^*$.




no reward given: only states and the expert's actions

| QUESTION | ASKS |
|---|---|
| **Behaviour cloning** | Can we learn the expert's policy directly, with supervised learning? |
| **Inverse RL** | Can we recover the reward $R$? |
| **Apprenticeship learning** | Can we use that $R$ to find a good policy? |

# Behaviour cloning

- The simplest answer: treat it as **supervised learning**. Every state the expert visited is an input; the action it took is the label.

$$\hat{\pi} = \arg\min_{\pi} \sum_{(s,a)\in D} -\log \pi(a \mid s)$$

- Any classifier works: a table, a decision tree, a neural network.
- It needs **no dynamics model**: you never ask what happens next, only what the expert did here.
- Nothing in the loss knows about time: each pair is treated as independent.



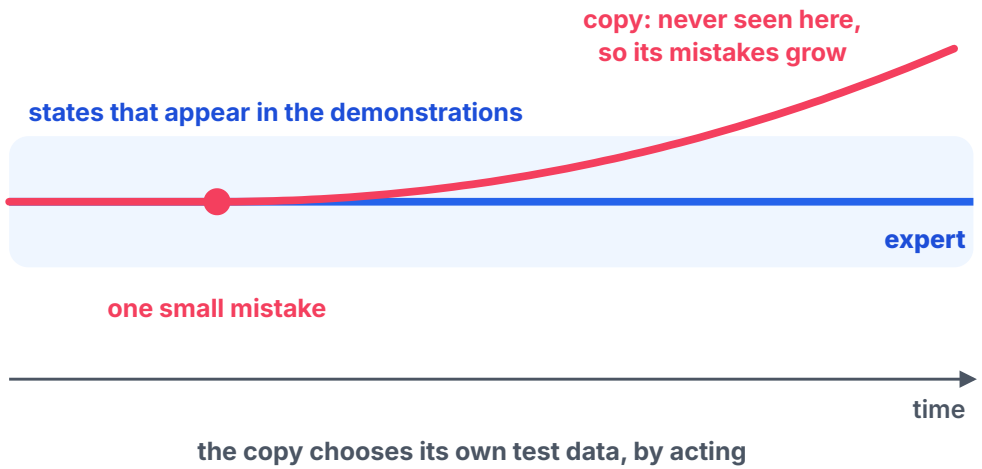
IN PLAIN WORDS

This is the moment in the course where the supervised learning people finally relax.

TRY IT

Demonstrations at a crossing: in “light red” the expert stopped 48 times and went twice; in “light green” it went 50 times. A table classifier picks the most frequent label: stop on red, go on green.

# Compounding errors




Our sketch of the idea, not data.

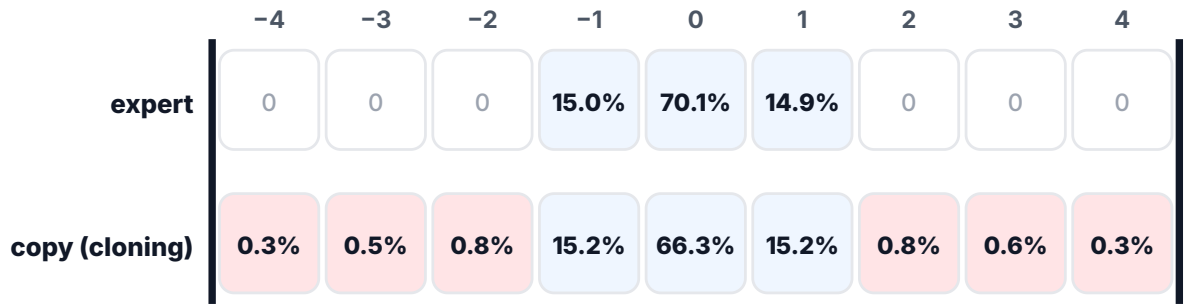
- A good expert rarely makes mistakes, so the data **never shows how to recover** from one.
- The copy makes one small error and lands in a state the expert never visited. There it has no idea what to do, so it makes a bigger error, and another.
- In supervised learning, test data comes from the same distribution as the training data. Here the copy **chooses its own test data**, by acting: **distribution shift**.



**IN PLAIN WORDS**

Imagine learning to drive only from videos of perfect drivers. The first time you drift, you're on your own.

# Our run: keeping a car in its lane




share of all time steps in each cell · thick lines: the road's edges
blue cells: the only cells in the demonstrations

| POLICY | DRIVES OFF THE ROAD |
|---|---|
| **Expert** | 0% of 2,000 drives |
| **Behaviour cloning** | **36.5%** of 2,000 drives |

- A road 9 cells wide. A gust pushes the car one cell left or right, 15% of the time each.
- The expert always steers back towards the centre. In 20 demonstrations it never left the middle three cells.
- The copy: a table built from those demonstrations. 5% of the time it acts at random; in cells with no data it always does.
- Each drive lasts 200 steps.



**IN PLAIN WORDS**

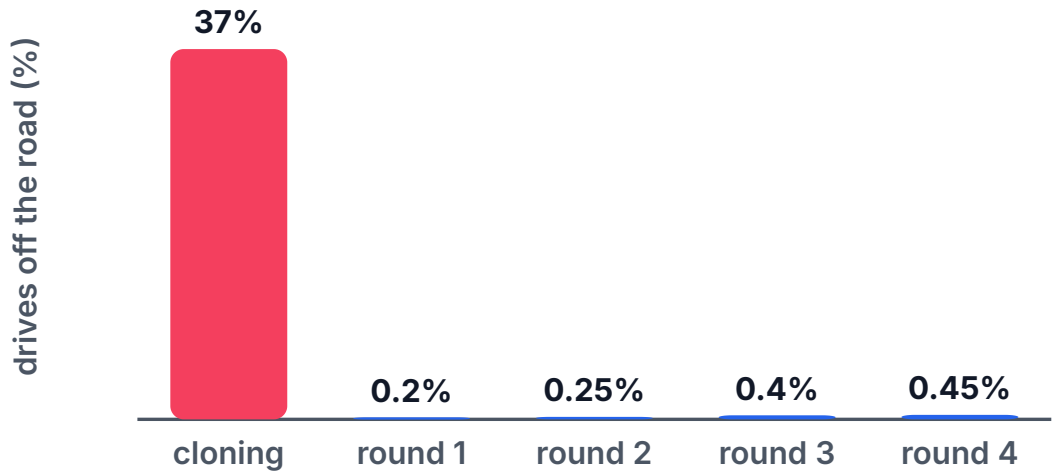
The copy is perfect wherever it has data. Its trouble starts in the pink cells, where it has none.

# DAgger: dataset aggregation

**DAGGER (ROSS, GORDON & BAGNELL, 2011)**

1. Train $\pi_1$ on the expert's demonstrations $D$.
2. For $i = 1, 2, \ldots$: run $\pi_i$ and record the states it visits.
3. Ask the expert what it would have done in each of those states: $\pi^*(s)$.
4. Aggregate: $D \leftarrow D \cup \{(s, \pi^*(s))\}$; train $\pi_{i+1}$ on all of $D$.

- Now the data contains the copy's own mistakes, with the right correction attached.
- The price: the expert must label states **every round**. Experts are expensive, and they get bored.




Our run: the lane toy, 20 learner drives per round, 2,000 test drives per point. After four rounds the data covers cells −3 to 4; the differences between rounds 1 to 4 (4 to 9 crashes out of 2,000) are within noise.

# Why the drift matters

- Let $\varepsilon$ be the copy's mistake rate on the states it is trained on, and $T$ the horizon.
- Behaviour cloning: the extra cost from small mistakes can grow like $T^2\varepsilon$ (Ross & Bagnell, 2010). One early mistake can ruin every step that follows.
- DAgger: roughly $T\varepsilon$, under its assumptions (Ross et al., 2011), because it trains on the states the copy actually visits.

| METHOD | NEEDS THE DYNAMICS? | EXPERT HELP AFTER THE DEMOS? | COST OF SMALL MISTAKES |
|---|---|---|---|
| **Behaviour cloning** | no | no | up to $\sim T^2\varepsilon$ |
| **DAgger** | no | yes, every round | $\sim T\varepsilon$ |

**TRY IT**

A 1% mistake rate over $T = 100$ steps. $T^2\varepsilon = 100$, but $T\varepsilon = 1$: a hundredfold difference in the bounds, from the same per-step accuracy.



**IN PLAIN WORDS**

Warm-up, settled: (a) false, cloning needs no dynamics. (b) false, DAgger asks the expert again every round.

# Check yourself: Part 1

CHECK YOURSELF

1. Why doesn't behaviour cloning need a dynamics model?

CHECK YOURSELF

2. In the lane toy, the copy matched the expert in every state it had data for. Why did it still drive off the road in over a third of the drives?

CHECK YOURSELF

3. With a mistake rate $\varepsilon = 0.5\%$ and horizon $T = 200$, compare $T^2\varepsilon$ and $T\varepsilon$.





HINT

Answers are at the end of the deck.

Applied Computational Mathematics Lab

PART 2

# Inverse RL

Don’t copy the actions; find what the expert wanted.



# Why recover the reward?

- Inverse RL: from demonstrations, recover a reward $R$ that the expert seems to be optimising.
- A reward **transfers**. New car, new road, new weather: the right actions change, the goal of not crashing does not.
- A common choice: a reward that is **linear in features** of the state,

$$R(s) = w^\top x(s), \qquad x(s) \in \mathbb{R}^k$$

- e.g. speed, distance to the car ahead, offset from the lane centre. Then learning $R$ means learning $k$ weights.
- Apprenticeship learning is then the easy part: any method from this course can find a policy for the recovered $R$.

**EXAMPLE**

A cloned driver trained on dry roads has never seen snow. A recovered reward that says “stay in the lane, keep a safe gap” still makes sense on snow; plan or learn with it there.



**IN PLAIN WORDS**

Copying says what the expert did. Inverse RL asks why.

# Feature matching

- Discounted feature expectations of a policy:

$$\mu(\pi) = \mathbb{E}\Big[\sum_{t\geq 0}\gamma^t x(S_t)\,\Big|\,\pi\Big] \quad\Longrightarrow\quad V^\pi = w^\top \mu(\pi)$$

- Want a reward under which the expert outperforms other policies (Abbeel & Ng, 2004).
- For a policy $\pi$ to do as well as $\pi^*$, it is sufficient that its feature expectations match. If $\|\mu(\pi) - \mu(\pi^*)\|_1 \leq \epsilon$, then for all $w$ with $\|w\|_\infty \leq 1$ (Hölder's inequality):

$$\left|w^\top \mu(\pi) - w^\top \mu(\pi^*)\right| \leq \epsilon$$

**TRY IT**

Two features. $\mu(\pi^*) = (3.0, 0.4)$, $\mu(\pi) = (2.9, 0.5)$: the $L_1$ gap is 0.2. Whatever the true weights in $[-1, 1]$, the values differ by at most 0.2. The worst case is $w = (1, -1)$: $2.6$ vs $2.4$.



**IN PLAIN WORDS**

Visit the same features as often as the expert, and you earn about the same, whatever the weights are.

# The catch: ambiguity

- There are **infinitely many rewards** with the same optimal policy.
- Some are trivial:
  - $R = 0$ everywhere makes **every** policy optimal;
  - scaling, $R$ and $10R$, changes nothing about which policy is best.
- There are also infinitely many **stochastic policies** that match the expert's feature counts.
- Which reward, and which policy, should we choose?

**EXAMPLE**

An expert walks around a pond to reach a bench. Did they avoid the pond because water is bad, or because the path on that side is nicer, or did they just like that route? All three rewards explain the same walk.



**IN PLAIN WORDS**

We need a principle for picking one explanation. Next part: commit to nothing you don't actually know.

Applied Computational Mathematics Lab

PART 3

# Maximum entropy IRL

Of all the explanations, pick the least committed one.

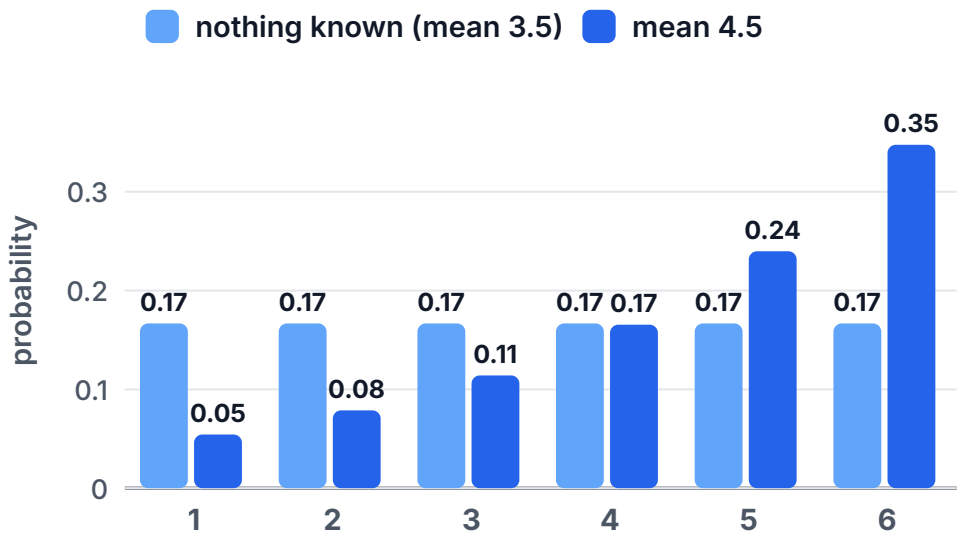


# The principle of maximum entropy

- Entropy of a distribution: $H(p) = -\sum_x p(x) \log p(x)$. It measures how spread out, how uncommitted, $p$ is.
- Principle (Jaynes, 1957): the distribution that best represents what you know is the one with the **largest entropy**, among those consistent with the constraints you know.
- Consider all distributions consistent with the data; pick the most spread out.

**TRY IT**

A die, and nothing else known: the answer is $1/6$ per face, $H = \log 6 \approx 1.79$. Now add one fact: the average roll is 4.5. Maximum entropy gives $p(k) \propto e^{\lambda k}$, with $\lambda = 0.371$.




Our computation. With mean 4.5 the entropy is 1.61. Putting all the weight on 4 and 5 also has mean 4.5, but entropy only 0.69: it claims a 1 is impossible, which nobody told us.

# Maximum entropy for IRL

- Ziebart et al. (2008) apply the same idea to **trajectories**. What we know: the expert's demonstrations $D$, summarised by their features $\mu(\tau)$. With a linear reward:

$$\max_P \; -\sum_\tau P(\tau)\log P(\tau) \quad \text{s.t.} \quad \sum_\tau P(\tau)\,\mu(\tau) = \frac{1}{|D|}\sum_{i\in D}\mu(\tau_i), \qquad \sum_\tau P(\tau) = 1$$

- More generally, find a trajectory distribution $p(\tau)$ with the same expected reward $r_\phi$ as the demonstrations' empirical distribution $\hat{P}(\tau)$, and maximum entropy otherwise.

$$\max_p \; -\sum_\tau p(\tau)\log p(\tau) \quad \text{s.t.} \quad \sum_\tau p(\tau)\,r_\phi(\tau) = \sum_\tau \hat{P}(\tau)\,r_\phi(\tau), \qquad \sum_\tau p(\tau) = 1$$

- To do it: alternate between fitting the reward, finding a policy for it, and updating the state frequencies the reward needs.
- Note: the original paper assumed the dynamics model is known.



**IN PLAIN WORDS**

Match what the expert does, on average. Beyond that, stay as uncommitted as possible.

# From maximum entropy to a distribution over trajectories

- Solving the constrained problem (Lagrange multipliers) always gives the same shape, an **exponential family**:

$$P(\tau \mid \phi) = \frac{1}{Z(\phi)} \exp\left(r_\phi(\tau)\right), \qquad Z(\phi) = \sum_\tau \exp\left(r_\phi(\tau)\right)$$

- Better trajectories are **exponentially** more likely, but none is impossible: the expert is assumed to be good, and a little noisy.
- Two trajectories with equal reward get equal probability: the expert has no favourite, so neither do we.
- For deterministic dynamics as written; with stochastic dynamics the transition probabilities multiply in.
- Same exponential as the softmax policies of Lecture 10, one level up: over whole trajectories.

**TRY IT**

Three trajectories with rewards 2, 1 and 1. $Z = e^2 + 2e \approx 12.83$, so the probabilities are $e^2/Z \approx 0.58$ and $e/Z \approx 0.21$ each. A reward higher by 1 makes a trajectory $e \approx 2.7$ times as likely.



**IN PLAIN WORDS**

Maximising entropy subject to matching features gives the same answer as maximising the likelihood of the data under this exponential family (Jaynes, 1957). Two roads, one answer.

# Maximum likelihood: from trajectories to states

$$J(\phi) = \sum_{\tau_i \in D} \log P(\tau_i \mid \phi) = \sum_{\tau_i \in D} r_\phi(\tau_i) - |D| \log Z(\phi) \quad \Longrightarrow \quad \nabla_\phi J = \sum_{\tau_i \in D} \nabla_\phi r_\phi(\tau_i) - |D| \sum_\tau P(\tau \mid \phi)\, \nabla_\phi r_\phi(\tau)$$

- If the reward of a trajectory is a sum over its states, $r_\phi(\tau) = \sum_{s \in \tau} r_\phi(s)$, the gradient becomes a sum over **states**:

$$\nabla_\phi J = \sum_s \left(\hat{\mu}_D(s) - |D|\, p(s \mid \phi)\right) \nabla_\phi r_\phi(s)$$

- Here $\hat{\mu}_D(s)$ counts the demonstrations' visits to $s$, and $p(s \mid \phi)$ is the expected number of visits per trajectory under the current policy.
- For a linear reward, $\nabla_\phi r_\phi(s) = f_s$, the features of $s$.

**STATE DENSITIES (KNOWN DYNAMICS, HORIZON $T$)**

1. $\mu_1(s) = p(S_0 = s)$.
2. $\mu_{t+1}(s') = \sum_s \sum_a \mu_t(s)\, \pi(a \mid s)\, P(s' \mid s, a)$.
3. $p(s \mid \phi) = \sum_{t=1}^{T} \mu_t(s)$.



**IN PLAIN WORDS**

More of what the expert does; less of what I do and the expert doesn't. I wish my own habits had a gradient this simple.

# Maximum entropy IRL: the algorithm

**MAXIMUM ENTROPY IRL (KNOWN DYNAMICS, LINEAR REWARDS)**

1. Input: expert demonstrations $D$.
2. Initialise $r_\phi$.
3. Compute the (soft) optimal $\pi(a \mid s)$ for $r_\phi$, e.g. with value iteration.
4. Compute the state visitation frequencies $p(s \mid \phi, T)$.
5. Compute the gradient: $\nabla_\phi J = \frac{1}{|D|} \sum_{\tau_i \in D} f_{\tau_i} - \sum_s p(s \mid \phi, T)\, f_s$.
6. Update $\phi$ with one gradient step.
7. Go to step 3.

- Notice what the gradient doesn't need: the expert's exact actions. Only where the expert went, and how often.

**BEFORE YOU READ ON**

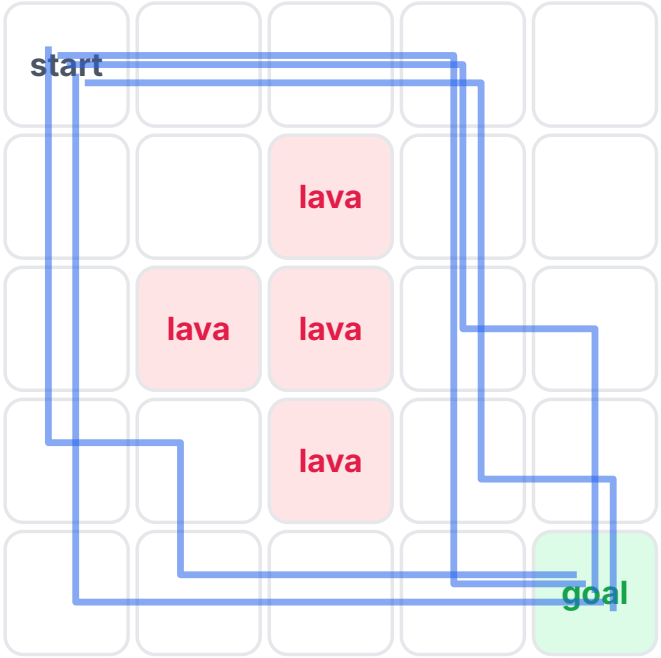
Which steps rely on knowing the dynamics model? (1) computing the optimal policy; (2) computing the visitation frequencies; (3) computing the gradient; (4) none.



**IN PLAIN WORDS**

In a big world, computing those visitation frequencies exactly is expensive: one reason later methods sample instead.

# Our run: max-entropy IRL around a lava lake




True world, and 5 of the 30 demonstrations (blue).

| | | | | |
|---|---|---|---|---|
| −0.9 | −0.2 | 0.5 | 1.5 | 0.3 |
| −0.8 | −1.7 | −2.3 | 0.5 | 0.4 |
| −0.1 | −1.7 | −1.2 | 0.0 | 0.1 |
| 0.9 | 0.2 | −1.6 | −0.3 | −0.1 |
| −0.3 | 1.1 | 0.4 | 0.5 | 5.0 |

Learned reward per cell (centred); lava outlined.

- A 5×5 grid, 12 steps per episode, start top-left, goal bottom-right. Lava costs −1, every other step −0.1.
- Expert: near-optimal under the true reward; 30 demonstrations. The learner sees only the paths: **nobody tells it about the lava**. It learns one reward per cell.
- 300 updates with known dynamics. The gap between its visit counts and the expert's ($L_1$) shrank from 15.6 to 0.18.

# What our learner found




| POLICY (2,000 RUNS) | REACHES GOAL | STEPS ON LAVA |
|---|---|---|
| **Expert** | 100% | 0% |
| **Learned, max-ent** | 99.95% | 5.2% |
| **Random** | 0.45% | 49.1% |

- The learned policy reaches the goal almost every time. It still steps on lava in about 5% of runs: a maximum-entropy policy stays a little noisy, and the near-optimal expert never did.
- The learned reward resembles the true one, but isn't it: correlation 0.82.
- It can't tell lava from a cell the expert simply never visited: cell (1, 1) is dry, but gets −1.7, as low as the lava.



**IN PLAIN WORDS**

Many rewards explain the same paths. Max-ent picks a reasonable one, not the true one.

# Beyond the original method

- Maximum entropy IRL has been hugely influential.
- Ziebart et al. (2008) assumed a **linear** reward and a **known dynamics model**. Was either needed in behaviour cloning? No.
- Finn et al. (2016), guided cost learning: general reward functions, learned while the robot acts, **without** knowing the dynamics.
- Ho & Ermon (2016), generative adversarial imitation learning (GAIL): a critic tries to tell expert from learner, and the learner tries to fool it.
- Imitation can greatly reduce the data needed to learn a good policy; combining it with online RL is an open, exciting area (for theory, see Sun et al., 2017).



IN PLAIN WORDS

GAIL is a lot like the adversarial networks that generate fake photos. Only here, the fakes are behaviour.

WHAT YOU SHOULD KNOW

Define behaviour cloning and how it differs from RL. Explain the principle of maximum entropy, the trajectory distribution it gives, and how it is used to learn a reward and then a policy.

# Check yourself: Parts 2 and 3

CHECK YOURSELF

4. Give two different rewards that make the same policy optimal.

CHECK YOURSELF

5. $\mu(\pi^*) = (2, 1)$ and $\mu(\pi) = (2.1, 0.7)$. For weights in $[-1, 1]$, what is the largest possible value gap?

CHECK YOURSELF

6. Under the max-ent distribution, three trajectories have rewards 3, 3 and 1. What are their probabilities?

CHECK YOURSELF

7. Which steps of max-entropy IRL need the dynamics model?

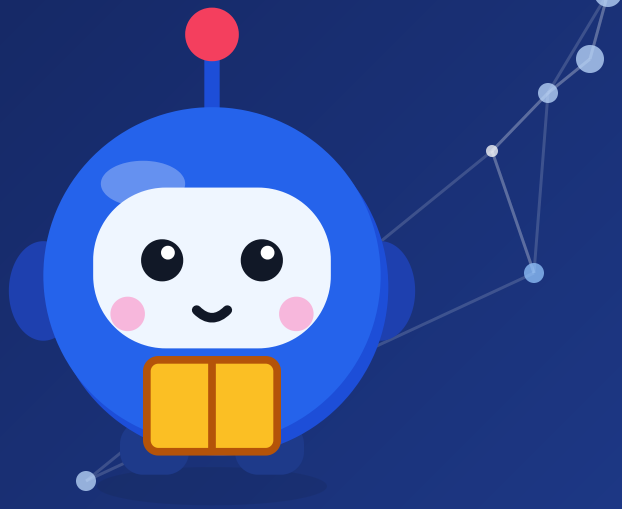


Applied Computational Mathematics Lab

PART 4

# Learning from comparisons

When people can’t demonstrate, they can still choose.



# Many ways for people to help

| FEEDBACK | WHAT IT SAYS | EXAMPLE |
|---|---|---|
| **Demonstration** | what to do | the driving examples of Part 1 |
| **Rewards while acting** | good or bad, right now | TAMER (Knox & Stone, 2008) learns a model of a person's rewards, like a clicker for a dog |
| **Guidance** | what to try next | Thomaz & Breazeal (2008): people teaching a robot used rewards to guide future actions, not only to grade past ones |
| **Correction** | not that; this instead | DAgger's expert labels |
| **Comparison** | this one is better | which of two clips, answers or rankings is better |






IN PLAIN WORDS

Each kind of feedback costs a different amount of human effort, and carries different information. The more each label costs, the fewer labels you get: experts have day jobs.

# Why pairs?

- Pairwise comparisons are often easier for people than hand-writing a reward function.
- And easier than giving a **scalar** score: “how much do you like this ad, from 1 to 10?”
- Absolute ratings are noisy, and every person uses the scale differently. Comparisons don’t care whether someone is generous or strict with numbers.
- Where it’s used:
  - comparing recommendation or ranking systems by showing people mixed results (dueling bandits);
  - active preference learning for robots: ask the comparison that teaches the most (Sadigh et al., 2017).



**IN PLAIN WORDS**

Ask me to rate a pizza from 1 to 10, I’ll say 7. Every pizza is a 7. Put two pizzas in front of me, and suddenly I have very strong opinions.

**EXAMPLE**

Two labellers rate the same answer 4 and 8. Both prefer it to a second answer, which they rated 2 and 6. Their scales disagree; their comparisons agree.

# The Bradley–Terry model (1952)

- Without more assumptions, the latent reward is not unique. So pick a structural model of how people choose.
- Start simple: $K$ options $b_1, \ldots, b_K$, no state (like a $K$-armed bandit). Each option has a hidden score $r(b_i)$. A person prefers $b_i$ to $b_j$ with probability

$$P(b_i \succ b_j) = \frac{\exp(r(b_i))}{\exp(r(b_i)) + \exp(r(b_j))} = \sigma\big(r(b_i) - r(b_j)\big) = p_{ij}$$

- Only **differences** matter: add the same number to every score and nothing changes.
- Transitive: $p_{ik}$ is determined by $p_{ij}$ and $p_{jk}$, since $\operatorname{logit} p_{ik} = \operatorname{logit} p_{ij} + \operatorname{logit} p_{jk}$.

**TRY IT**

Scores 1, 0 and −1. Option 1 beats option 2 with probability $\sigma(1) \approx 0.73$; it beats option 3 with $\sigma(2) \approx 0.88$. Equal scores: a coin flip.



**IN PLAIN WORDS**

Like chess ratings: the rating difference between two players tells you how likely one is to beat the other.

# Who wins?

THREE DEFINITIONS

**Condorcet winner**: $b_i$ with $P(b_i \succ b_j) > 0.5$ for every other $b_j$.
**Copeland winner**: the most pairwise victories (score: items it beats minus items it loses to).
**Borda winner**: the highest average probability of beating the others.

| ROW BEATS | A | B | C | D | BORDA | COPELAND |
|---|---|---|---|---|---|---|
| **A** | – | 0.55 | 0.55 | 0.55 | **0.55** | **3** |
| **B** | 0.45 | – | 0.95 | 0.95 | **0.78** | 1 |
| **C** | 0.45 | 0.05 | – | 0.60 | 0.37 | −1 |
| **D** | 0.45 | 0.05 | 0.40 | – | 0.30 | −3 |

Each row: the probability that the row option beats the column option. Our made-up preferences, not from Bradley–Terry.

TRY IT

A narrowly beats everyone: the Condorcet and Copeland winner. B crushes C and D: the best average, so the Borda winner. Which you want depends on the application.

- Under Bradley–Terry, the option with the highest score is all three at once.
- Historically, algorithms for $K$-armed dueling bandits mostly looked for a Copeland winner.

# Fitting the scores

- Data: $N$ tuples $(b_i, b_j, \mu)$, with $\mu = 1$ if the person preferred $b_i$, $0.5$ for a tie, $0$ if they preferred $b_j$.
- Maximise the likelihood, i.e. minimise the **cross-entropy**:

$$\mathcal{L}(r) = -\sum_{(b_i, b_j, \mu) \in D} \Big[\mu \log P(b_i \succ b_j) + (1-\mu) \log P(b_j \succ b_i)\Big]$$

- It is logistic regression on differences of scores.
- Each comparison nudges the winner's score up and the loser's down, by $1 - p$: more when the result was a surprise.
- Two options never linked by any chain of comparisons? Their gap can't be learned.

**TRY IT**

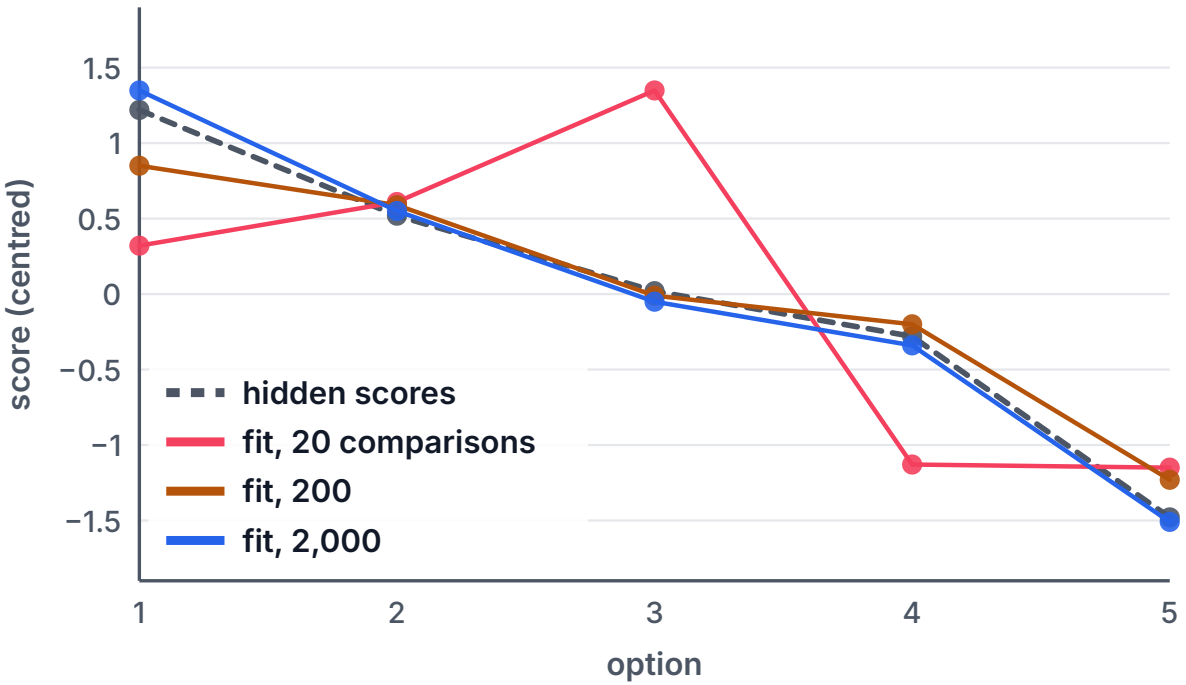
Scores start equal, $r_i = r_j = 0$, and $b_i$ wins: $p = 0.5$, so the gradient step with rate 0.2 moves $r_i$ up by 0.1 and $r_j$ down by 0.1. Had $b_i$ already been far ahead, $p \approx 1$, and almost nothing moves.



**IN PLAIN WORDS**

Each comparison is a tiny classification problem: did $i$ win, or did $j$?

# Our run: Bradley–Terry from noisy comparisons




Our run: 5 options; a simulated person compares random pairs and chooses by Bradley–Terry. Fitted by gradient descent on the cross-entropy.

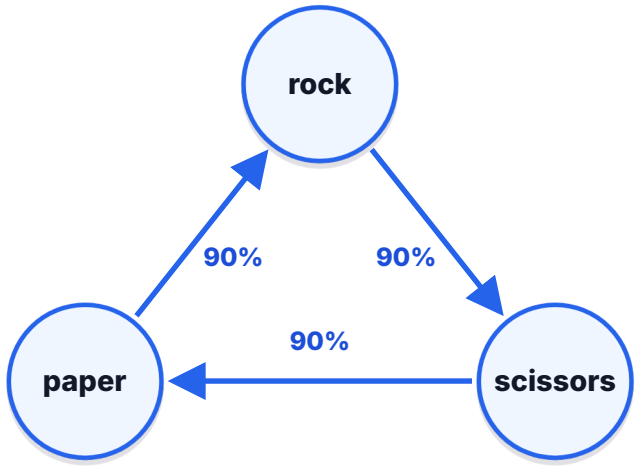
- After 20 comparisons the ranking is wrong: option 3 comes out on top. Twenty noisy answers are not much.
- After 200 the ranking is exactly right.
- After 2,000 the scores are close to the hidden ones; the largest error is 0.13, on the top option.



**IN PLAIN WORDS**

More comparisons, of more different pairs, and the fitted scores approach the hidden ones.

# When one score per option isn’t enough




each arrow: “beats”, 90% of the time

- A person with a cycle: rock beats scissors, scissors beats paper, paper beats rock, each 90% of the time.
- We fitted Bradley–Terry to 1,200 such comparisons. All three scores came out at about zero: −0.02, 0.03, −0.01.
- So the model predicts a coin flip for every pair (0.498); the truth is 0.9, every time.
- One score per option is transitive by construction: it simply can’t hold a cycle.



**IN PLAIN WORDS**

Do real people have cycles? Ask anyone choosing a restaurant with friends. It gets worse when you combine many people: hold that thought for Part 6.

# Check yourself: Part 4

CHECK YOURSELF

8. Under Bradley–Terry with scores 0.4 and −0.4, how often is the first option preferred?

CHECK YOURSELF

9. Under Bradley–Terry, $P(A \succ B) = 0.75$ and $P(B \succ C) = 0.75$. What is $P(A \succ C)$?

CHECK YOURSELF

10. Why are comparisons often more reliable than 1-to-10 ratings?



Applied Computational Mathematics Lab

PART 5

# RLHF

Reinforcement learning from human feedback: from backflips to chatbots.

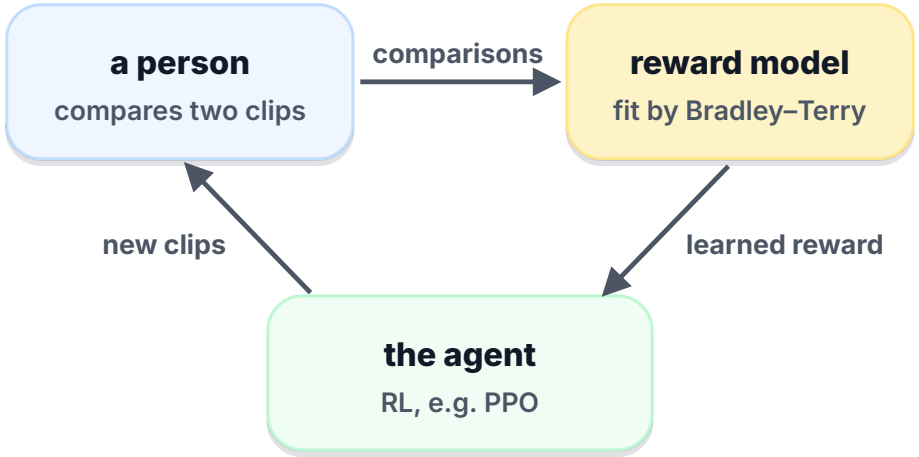


# From preferences to a reward model

- The same idea works for **trajectories**. Two trajectories $\tau^1, \tau^2$; let $R^1 = \sum_{t=0}^{T-1} r_t^1$ be the latent, unobserved sum of rewards of $\tau^1$, and similarly $R^2$.
- The probability that a person prefers $\tau^1$:

$$\hat{P}\left[\tau^1 \succ \tau^2\right] = \frac{\exp \sum_t r_t^1}{\exp \sum_t r_t^1 + \exp \sum_t r_t^2}$$

- Fit a reward model so that it agrees with the people. Then run RL on the **learned** reward, e.g. with PPO.
- The people teach the reward model; the reward model teaches the agent. The agent never talks to a person directly.




**LEARNING TO BACKFLIP**

Try writing a reward for a backflip: height? rotation speed? landing on your feet, not your face? Christiano et al. (2017) didn't. A person watched pairs of short clips and picked the one more like a backflip; the simulated robot learned to flip from about 900 bits of feedback (reported by Christiano et al., not our run).

# From backflips to chatbots

STAGE 1

### Pretraining

Predict the next word on a lot of text. Alone, this gives a model that continues text: ask it a question and it may answer with three more questions.

STAGE 2

### Instruction tuning

Fine-tune on demonstrations of good answers. An old friend: this is **behaviour cloning**.

STAGE 3

### Comparisons and RL

The model writes several answers to a prompt; people rank them. Fit a reward model (**Bradley–Terry**), then optimise against it with a **policy gradient** method, on a leash.

- The recipe used for InstructGPT (Ouyang et al., 2022) and, for dialogue, ChatGPT; many ChatGPT details, such as the data and model size, were never published.
- Every piece is from this course: supervised imitation, a preference model, and policy gradients (Lecture 10).



IN PLAIN WORDS

Three stages, three lectures' worth of ideas, at a very large scale.

# Reward models for language

- Human judgements are noisy and miscalibrated: one person's 7 is another's 4. So ask for **rankings** or pairwise comparisons, which are more reliable.
- A ranking of $K$ answers gives $K(K-1)/2$ comparisons. Train the reward model $RM_\phi$ with the Bradley–Terry loss:

$$J_{RM}(\phi) = -\mathbb{E}_{(s^w, s^l)\sim D}\Big[\log \sigma\big(RM_\phi(s^w) - RM_\phi(s^l)\big)\Big]$$

- $s^w$ is the winning answer, $s^l$ the losing one: push the winner up, the loser down. The reward model is often a copy of the language model with a new head that outputs one number.
- **Make sure your reward model works first**: test it on held-out human judgements. Stiennon et al. (2020) report that a large enough model, trained on enough comparisons, approached the agreement of a single human labeller (reported, not our run).

**OUR EXAMPLE**

News: "A storm closed the city's harbour for two days. No one was hurt." Three summaries: (1) "A two-day storm closed the harbour; no injuries." (2) "A storm caused huge damage at the harbour." (3) "The city has a nice harbour." A labeller ranks 1, 2, 3: that gives three training pairs, (1, 2), (1, 3) and (2, 3).

# RL with a KL price

- We have: a pretrained (possibly instruction-tuned) model $p^{PT}(s)$; a reward model $RM_\phi(s)$; and an RL method that can optimise any reward.
- Initialise a copy $p_\theta^{RL}(s)$ of the model, and optimise with RL:

$$r(s) = RM_\phi(s) - \beta \log \frac{p_\theta^{RL}(s)}{p^{PT}(s)}$$

- The second term is a price for wandering away from the pretrained model; in expectation it is the **KL divergence** between $p_\theta^{RL}$ and $p^{PT}$.
- Why pay it? The reward model was trained on answers the original model writes. Far from those answers, nobody checked it.

**TRY IT**

The new model makes an answer 4 times as likely as the pretrained model did. That answer pays $\beta \log 4 \approx 1.39\beta$. An answer it makes half as likely earns a bonus of $\beta \log 2$.



**IN PLAIN WORDS**

Optimise the reward, but on a leash. The leash length is $\beta$: keep it in mind.

# Does it work?

- Summarisation (Stiennon et al., 2020): RLHF gave summaries people preferred over pretraining plus supervised fine-tuning alone (reported).
- InstructGPT (Ouyang et al., 2022): labellers preferred the answers of a 1.3-billion-parameter model trained this way to those of the 175-billion-parameter GPT-3, a model more than 100 times larger (reported, not our run).
- Controlled comparisons (Dubois et al., 2023), often using a strong model's judgements as a stand-in for human feedback: PPO, the method behind InstructGPT, does work.
- But simple baselines work well too: **best-of-$n$**, and training on the good outputs only.



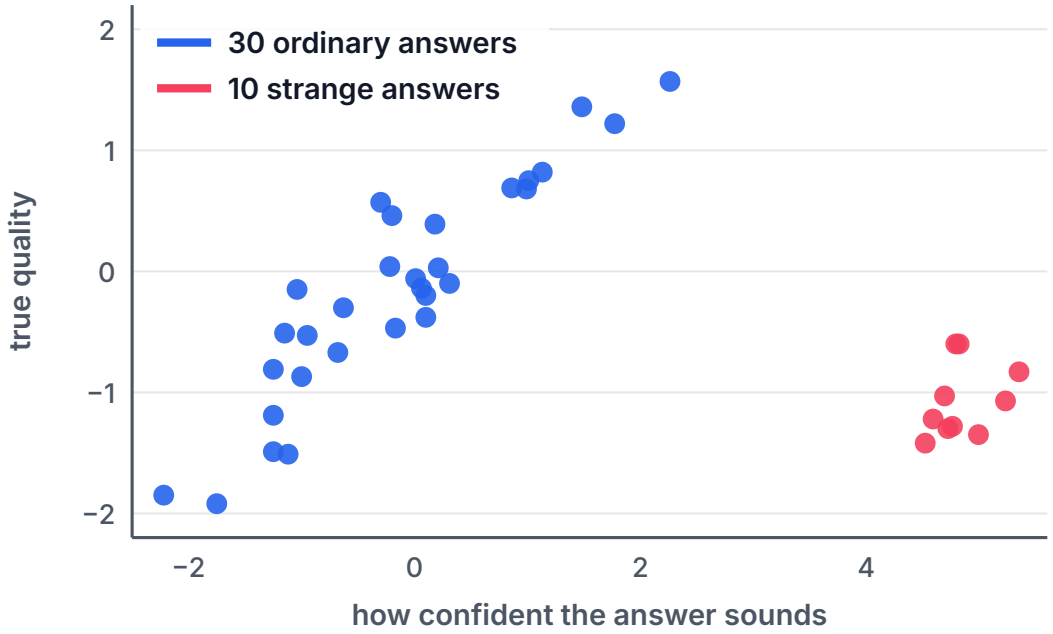
**IN PLAIN WORDS**

The same Bradley–Terry loss, the same policy gradients from Lecture 10. Just at a much larger scale.

**BEST-OF-N**

Sample $n$ answers from the model, and keep the one the reward model likes best. No RL at all. We'll come back to it in Part 6.

# Our toy RLHF: the setup




Our toy: 40 possible answers. Each point is one answer.

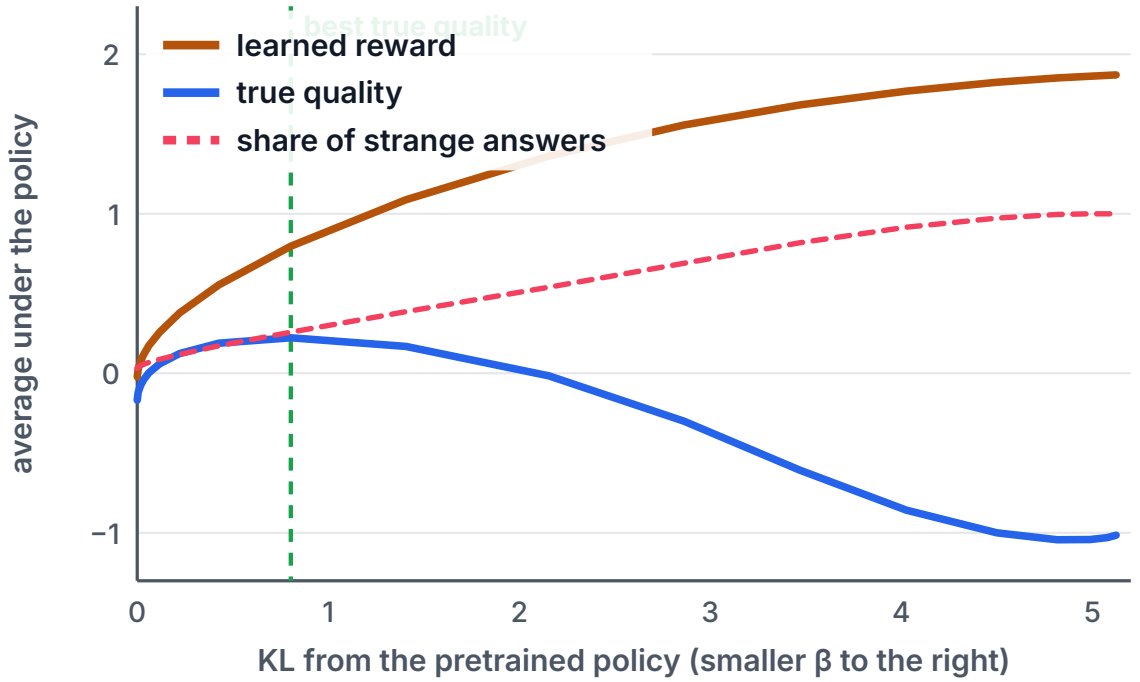
- A pretrained policy gives the 30 ordinary answers 97% of the time, and the 10 strange ones only 3%.
- The strange ones sound extremely confident, and they're wrong. The labeller only cares about quality (Bradley–Terry on true quality).
- The reward model can't see quality directly: it sees confidence plus a noisy hint of quality. On ordinary answers, the two go together.
- 3,000 comparisons later it agrees with the labeller on 81% of pairs; its weight on confidence is 0.32, on the hint 0.14.



IN PLAIN WORDS

It learned, sensibly: confident sounds good. On the answers it has seen, that is mostly true.

# Our toy RLHF: pulling the leash




Our run: the exact KL-regularised optimum, $\pi \propto p^{PT} e^{\hat{r}/\beta}$, for 25 values of $\beta$ from 32 down to 0.03.

- At first, all is well: the learned reward rises, and true quality rises with it, from −0.18 to 0.22, at a KL of about 0.8.
- Push further. The learned reward keeps climbing, to 1.87; true quality falls to −1.01.
- The policy found the confident, wrong answers: at the end it gives them practically every time. The reward model loved them; nobody had checked it there.



**IN PLAIN WORDS**

Every extra bit of optimisation after the best point made things worse. This is reward-model over-optimisation.

# Check yourself: Part 5

CHECK YOURSELF

11. The new model makes an answer 3 times as likely as the pretrained model did. With $\beta = 0.1$, what price does it pay?

CHECK YOURSELF

12. A ranking of 5 answers gives how many pairwise comparisons?

CHECK YOURSELF

13. In our toy, what would happen to the strange answers with $\beta = 0$? Why?



Applied Computational Mathematics Lab

PART 6

# The limits

What can go wrong when people write the reward.



PART 6 · LIMITS

# Reward hacking

- Optimise a learned reward too hard, and the policy finds where the reward model is wrong: **reward-model over-optimisation** (Stiennon et al., 2020; our toy in Part 5).
- A popular version of Goodhart's law: "when a measure becomes a target, it ceases to be a good measure". RLHF is a machine for turning measures into targets.
- For chatbots the danger has a specific shape: a model rewarded for responses that **seem** authoritative and helpful can learn to sound that way regardless of truth, and then make things up, confidently.



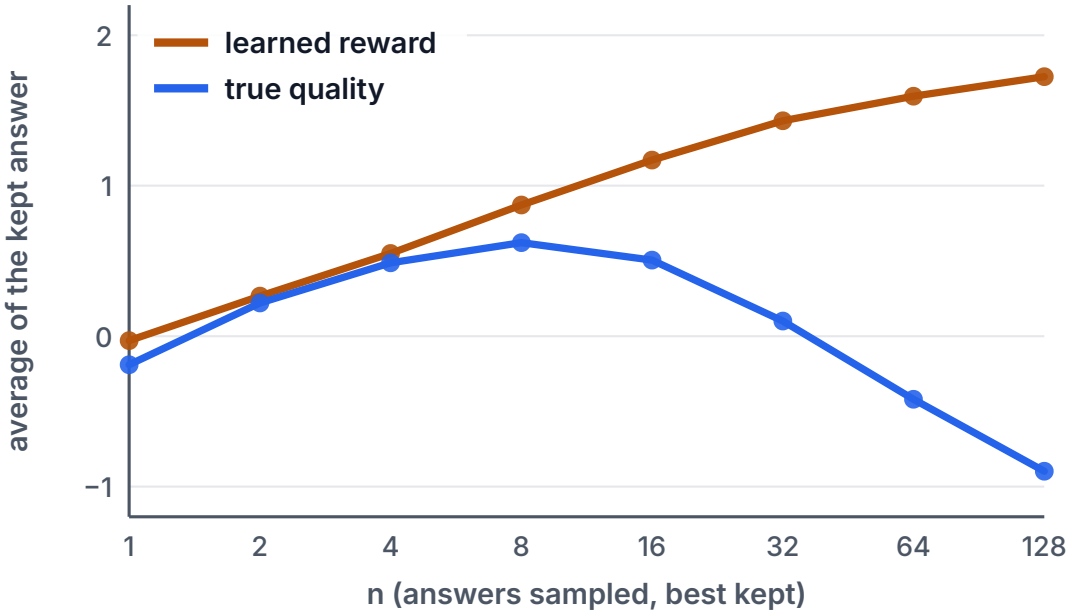
IN PLAIN WORDS

The KL leash, careful evaluation with real people, and more and better comparisons all help. None makes the problem go away.

EXAMPLE

An essay grader that likes long words gets essays full of long words. A reward model that likes confident answers gets confident answers. Neither asked for correct ones.

# Best-of-n on our toy




Our run: the toy of Part 5, 4,000 draws per point.

- No RL: sample $n$ answers from the pretrained policy, keep the one the reward model scores highest.
- Best of 8 was the sweet spot: true quality +0.62. Best of 128: −0.90, with a strange answer kept 94% of the time.
- Same lesson, different tool: search hard enough against a flawed reward and you find its flaws.



**IN PLAIN WORDS**

Best-of-n has a natural knob, $n$. Like $\beta$, turn it up slowly, and keep checking with real people.

# Human preferences are unreliable

- People get tired, and rushed. People like longer answers, and nicer formatting.
- Dubois et al. (2023): after RLHF-style training, answers became much more detailed, with nicer, list-like formatting. Is a longer answer a better one? Sometimes, not always.
- And a **model** of human preferences is even less reliable than the humans themselves.

**WHO LABELS**

Where do the labels come from? The CS234 lecture notes that RLHF labels are often obtained from low-wage workers overseas. Both the quality and the fairness of that work matter.

- Whose opinions end up in the model depends on who labelled the data. Santurkar et al. (2023) asked whose opinions language models reflect, and found they match some groups far better than others.



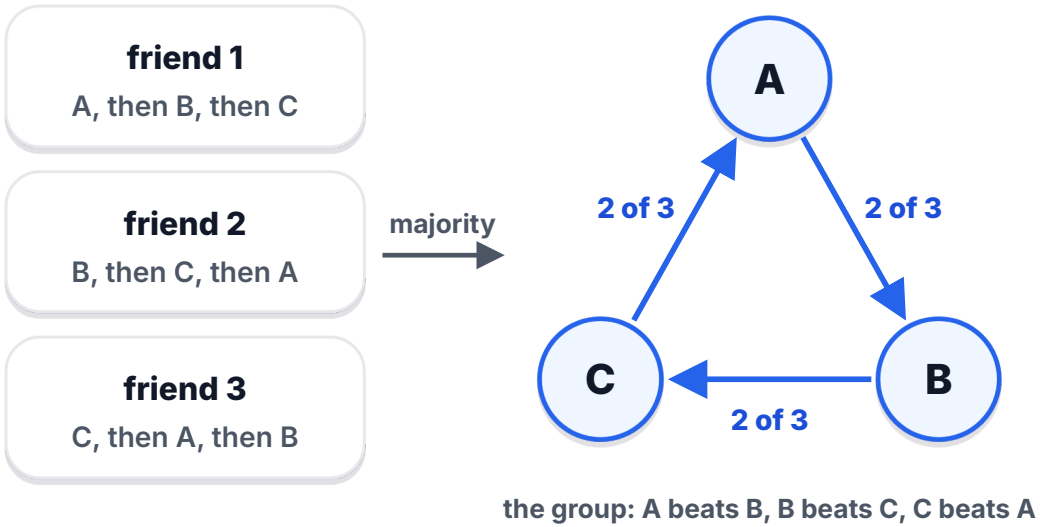
**IN PLAIN WORDS**

Be honest: have you ever upvoted an answer because it looked thorough, without checking it? Long and wrong still looks impressive at first glance.

**IN PRACTICE**

None of this is a reason to stop. It is a reason to report who labelled your data, and how, as carefully as you report your learning rate.

# When a group disagrees




Our example of Condorcet's paradox.

**TRY IT**

Count the votes. A above B: friends 1 and 3. B above C: friends 1 and 2. C above A: friends 2 and 3. Every pair is won 2 to 1, and the wins go round in a circle.

- Three friends, three restaurants, three perfectly consistent rankings.
- Take a majority vote on each pair: A beats B, B beats C, and C beats A. The group has a cycle, though nobody is irrational.
- Recall Part 4: one Bradley–Terry score per option can't hold a cycle.
- When people genuinely disagree, a single reward model must pick a side, or average them into something nobody actually wants.



**IN PLAIN WORDS**

Learning from human preferences sits where social choice, computational economics and AI meet. One reward model for everyone is a strong assumption.

# Skipping the reward model: DPO

- Direct preference optimisation (Rafailov et al., 2023): no separate reward model, and no RL loop. Optimise the policy on the comparisons directly:

$$\mathcal{L}_{DPO}(\theta) = -\mathbb{E}_{(x,y_w,y_l)}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$$

- Under the hood: the same Bradley–Terry model and the same KL leash. The term $\beta \log \frac{\pi_\theta}{\pi_{\text{ref}}}$ plays the role of the reward.
- It still learns from human preferences, so the questions about labels and labellers don't go away.
- To learn more: a new Stanford course on this area, CS329H, Machine Learning from Human Preferences (Koyejo).



**IN PLAIN WORDS**

Fewer moving parts. The same old question: whose preferences, and how carefully were they collected?

**TRY IT**

Prompt $x$, winner $y_w$, loser $y_l$. If $\pi_\theta$ has made $y_w$ twice as likely as $\pi_{\text{ref}}$ does, and left $y_l$ unchanged, the argument of $\sigma$ is $\beta \log 2$: with $\beta = 1$, the loss is $-\log \sigma(0.69) \approx 0.40$.

# Check yourself: Part 6

CHECK YOURSELF

14. In our best-of-n toy, the learned reward of the kept answer keeps rising with $n$. Why does its true quality fall?

CHECK YOURSELF

15. Name two reasons human preference labels can be unreliable.

CHECK YOURSELF

16. Three voters rank X, Y, Z as: X, Y, Z / X, Z, Y / Z, Y, X. Does the majority have a cycle? Who wins each pair?



WRAP-UP

# Summary

- **Imitation learning**: learn from demonstrations when showing is easier than writing a reward.
- **Behaviour cloning** is supervised learning; it drifts, since the copy picks its own test states. **DAgger** fixes the drift, at the price of an expert every round.
- **Inverse RL** recovers a reward; many rewards explain the same behaviour. **Maximum entropy** picks the least committed one: $P(\tau) \propto e^{r(\tau)}$, and its gradient matches visit counts.
- **Comparisons** are cheap and consistent; **Bradley–Terry** turns them into scores, but can't hold cycles.
- **RLHF**: a reward model from comparisons, then RL with a KL leash. Pull the leash too hard and it optimises the wrong thing.
- Labels are noisy and come from real people; groups don't always agree. DPO and best-of-$n$ are simpler, with the same questions.

$$\nabla_\phi J = \hat{\mu}_D - |D|\, p(s \mid \phi)$$

$$P(b_i \succ b_j) = \sigma(r_i - r_j)$$

$$r(s) = RM_\phi(s) - \beta \log \frac{p_\theta^{RL}(s)}{p^{PT}(s)}$$

WRAP-UP

# The whole course

LECTURE 1

**Introduction**

Agents, rewards, the reward hypothesis.

LECTURE 2

**Exploration and exploitation**

Bandits, regret, UCB.

LECTURE 3

**MDPs and dynamic programming**

Bellman equations, policy and value iteration.

LECTURE 4

**Theory of dynamic programming**

Contractions and convergence.

LECTURE 5

**Model-free prediction**

Monte Carlo and TD.

LECTURE 6

**Model-free control**

SARSA, Q-learning, off-policy.

LECTURE 7

**Function approximation**

Features, gradients, the deadly triad.

LECTURE 8

**Deep RL**

DQN and friends.

LECTURE 9

**Planning and models**

Dyna, search, MCTS.

LECTURE 10

**Policy gradients**

REINFORCE, actor-critics, trust regions.

LECTURE 11

**Learning about many things**

GVFs, PopArt, distributional RL.

LECTURE 12

**Learning from people**

Imitation, IRL, RLHF.



IN PLAIN WORDS

We started with a robot chasing one number: the reward hypothesis, any goal as one scalar. In this last lecture, people write that number. If you keep one slide from the whole course, keep the reward hypothesis, and the word in it that matters most: **hypothesis**.

ANSWERS

# Answers to check yourself

- **Warm-up.** (a) False. (b) False. (c) True.
- **1.** It only maps states to actions; it never asks what happens next.
- **2.** Noise and its 5% random actions took it to cells with no data, where it acted at random; errors compounded.
- **3.** $T^2\varepsilon = 200$, $T\varepsilon = 1$.
- **4.** E.g. $R$ and $2R$; or $R = 0$, which makes every policy optimal.
- **5.** $0.1 + 0.3 = 0.4$, reached by $w = (1, -1)$.
- **6.** $e^3/(2e^3 + e) \approx 0.47$ each, and $e/(2e^3 + e) \approx 0.06$.
- **7.** Computing the optimal policy, and the visitation frequencies.
- **8.** $\sigma(0.8) \approx 0.69$.
- **9.** $\text{logit} = \log 3 + \log 3 = \log 9$, so $9/10 = 0.9$.
- **10.** People use rating scales differently; comparisons don't depend on the scale.
- **11.** $0.1 \log 3 \approx 0.11$.
- **12.** $5 \times 4/2 = 10$.
- **13.** With no leash, the policy gives only the answer the reward model rates highest: a strange, wrong one (true quality −0.83 in our toy).
- **14.** With many samples, the kept answer is the one the reward model overrates most: the confident, wrong ones.
- **15.** E.g. fatigue, rushing, a taste for length or formatting, labellers' own opinions.
- **16.** No cycle: X beats Y and Z (2 to 1 each), and Z beats Y (2 to 1). X is a Condorcet winner.

CREDITS

# Sources and credits

- The order of topics follows *Lecture 8: Imitation Learning and RLHF* of Stanford CS234, Reinforcement Learning (Spring 2024), by Emma Brunskill, which itself uses material from Tatsu Hashimoto's CS224N lecture and Katerina Fragkiadaki's lecture on maximum entropy inverse RL. Part 1 also recaps behaviour cloning and DAgger from the previous CS234 lecture. This deck is not an official DeepMind, UCL or Stanford 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.
- Ross & Bagnell (2010); Ross, Gordon & Bagnell (2011), DAgger; Abbeel & Ng (2004); Jaynes (1957); Ziebart et al. (2008); Finn et al. (2016); Ho & Ermon (2016); Sun et al. (2017); Knox & Stone (2008); Thomaz & Breazeal (2008); Sadigh et al. (2017); Bradley & Terry (1952); Yue et al. (2012); Christiano et al. (2017); Stiennon et al. (2020); Ouyang et al. (2022); Dubois et al. (2023); Santurkar et al. (2023); Rafailov et al. (2023). Numbers marked "reported" are from these papers, not our runs.
- The lane, lava-grid, Bradley–Terry, rock–paper–scissors, toy RLHF and best-of-n runs, and the loaded die, are ours.
- Companion videos: **ACML Explains**, season 1, episodes 47–51. More from the lab: acml.teddylazebnik.com



Applied Computational Mathematics Lab

END OF LECTURE 12

# That’s the course

Twelve lectures, one reward hypothesis. Thank you for learning with us. Now go and write a good paper.

