Applied Computational Mathematics Lab

TEDDY LAZEBNIK'S COURSE · LECTURE 11

Reinforcement Learning

Lecture 11 · Learning about many things: general value functions, auxiliary tasks, trade-offs between tasks and PopArt, universal value functions, and distributional reinforcement learning



BEFORE WE START

Why learn about many things?

- Many deep RL algorithms optimise a very narrow objective.
- Narrow objectives induce narrow state representations, and narrow representations can't support good generalisation. The deadly triad and leakage (Lecture 8) only make it worse.
- Our agents should build rich knowledge about the world: learn about more than just the main task reward.

A SHIP

If the only thing you ever practised is sailing a ship in a straight line, you're lost the day someone asks for a circle. Better: first learn how a ship moves; then a straight line, a circle, whatever you're asked.



IN PLAIN WORDS

What knowledge should an agent learn? Two families of ideas today: general value functions (and their universal versions), and distributional value predictions.

BEFORE WE START

Today's map

PART 1

General value functions

Cumulants, discounts and target policies; predictive state.

PART 2

Auxiliary tasks

Extra predictions that shape a shared representation.

PART 3

Trade-offs and PopArt

When one loud task drowns the others, and how to rescale.

PART 4

Open problems

Off-policy GVFs, universal value functions, discovering what to predict.

PART 5

Distributional RL

Learn the whole distribution of returns: categorical and quantile.

READING

Papers

Sutton et al. (2011); Jaderberg et al. (2016); van Hasselt et al. (2016); Schaul et al. (2015); Bellemare et al. (2017); Dabney et al. (2017).



IN PLAIN WORDS

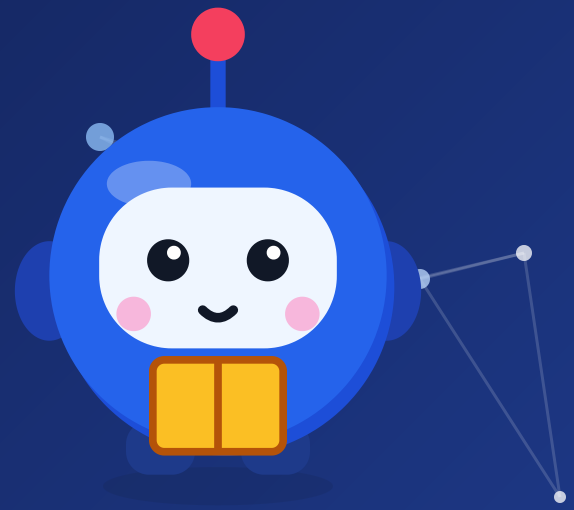
By the end you should be able to define a GVF by its three knobs, explain what PopArt preserves, and run one step of categorical TD by hand.

Applied Computational Mathematics Lab

PART 1

General value functions

Ask many questions, not just one.



The reward hypothesis

- All goals can be represented as maximisation of a scalar reward (Sutton & Barto, 2018).
- Then all useful knowledge may be encoded as **predictions about rewards**, or reward-like signals.
- For instance, in the form of **general value functions** (GVFs).



IN PLAIN WORDS

This course started very mathematical and slowly became more and more engineering. This is engineering: ask many questions about the world, and learn the answers with the same tools.

QUESTIONS

“How much reward is coming?” is one question. “How soon will I hit a wall?”, “Will the light turn red?” and “How far will I travel if I keep going forward?” are others. Each one is a value function.

General value functions

- A GVF is conditioned on more than states and actions (Sutton et al., 2011):

$$q_{c,\gamma,\pi}(s,a) = \mathbb{E}\left[C_{t+1} + \gamma_{t+1}C_{t+2} + \gamma_{t+1}\gamma_{t+2}C_{t+3} + \cdots \mid S_t = s, A_{t+i} \sim \pi(S_{t+i})\right]$$

- with $C_t = c(S_t)$ and $\gamma_t = \gamma(S_t)$.

KNOB	MEANING	LETS YOU
cumulant c : $\mathcal{S} \to \mathbb{R}$	what to add up	predict many things, including (but not only) reward
discount γ : $\mathcal{S} \to \mathbb{R}$	when to stop (termination)	predict over different horizons
target policy π	whose behaviour	predict under hypothetical policies



IN PLAIN WORDS

The policy knob: what would happen if I always walked left? You can learn that answer without ever walking left: off-policy learning.

Simple predictive questions

QUESTION	CUMULANT	DISCOUNT	POLICY
Policy evaluation	$C_t = R_t$	γ	the agent's π
Reward prediction	$C_t = R_t$	0	the agent's π
Next-state prediction	$\{C_t^i = S_t^i\}_i$ (or features $\phi^i(S_t)$)	0	the agent's π
"How soon will I hit the wall?"	1 at the wall, else 0	0.8, say	any π

- With $\gamma = 0$ a GVF predicts the very next signal; with γ near 1 it predicts a long-term sum.

FROM CLASS

A boxing robot only learned to win; it learned to fall over a lot, too. A new signal, "falling on the floor", changed the agent a lot. (In the evaluation, only winning counted.)

TRY IT

A "wall soon" GVF with $\gamma = 0.8$: if the wall is certainly hit in 3 steps and never before, its value is $0.8^2 = 0.64$.

Three questions, one stream of experience



- Our run: a random walk in a 7-cell corridor. Three GVFs learned at once by TD(0), $\alpha = 0.05$, from the same 20,000 steps.
- Cumulants: 1 on reaching cell 6 (“reward”); 1 on reaching cell 0 (“wall”).
- RMS error after 20,000 steps: 0.082, 0.009 and 0.035; exact values by solving the linear equations.



IN PLAIN WORDS

Same data, same TD rule, three different answers about the world.

Predictive state representations

- A large, diverse set of GVF predictions can be a sufficient statistic for any other prediction, including the value of the main task reward.
- In **predictive state representations** (Littman et al., 2002):
  - use the predictions themselves as the representation of state;
  - learn the policy and values as linear functions of these predictions.



IN PLAIN WORDS

Your state is no longer what you see. It's what you expect to happen.

TRY IT

A driver's "state" could be: will the light turn red within 5 seconds? Will the car ahead brake? How far to my exit? A linear policy on these three predictions already drives sensibly.

Check yourself: Part 1

CHECK YOURSELF

1. Name the three knobs of a GVF and what each controls.

CHECK YOURSELF

2. Which GVF predicts the next reward only?

CHECK YOURSELF

3. In our corridor, why is the “wall next step” prediction 0.5 in both cell 0 and cell 1?





HINT

Answers are at the end of the deck.

Applied Computational Mathematics Lab

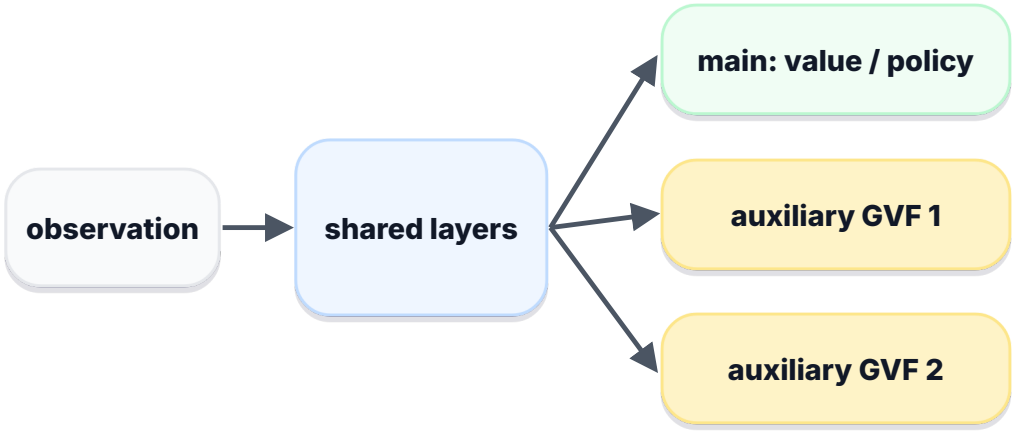
PART 2

Auxiliary tasks

Extra predictions that shape a shared representation.



GVFs as auxiliary tasks



one loss per head; all their gradients shape the shared layers

- GVFs can be used as auxiliary tasks (Jaderberg et al., 2016):
- they share part of the neural network;
- we minimise the main-task loss and the auxiliary GVF losses jointly;
- this forces the shared hidden layers to be richer and more robust.



IN PLAIN WORDS

You don’t really care about the auxiliary answers. You care about what they do to the shared layers.

Examples of auxiliary tasks

PIXEL CONTROL

UNREAL

Learn to change regions of the screen as much as possible: the agent learns what its actions affect.

FEATURE CONTROL

UNREAL

The same idea for the network's own hidden features: learn to change them as much as possible.

- In the UNREAL agent (Jaderberg et al., 2016) these auxiliary tasks made learning faster and the final agents better on many 3-D maze tasks.

FROM CLASS

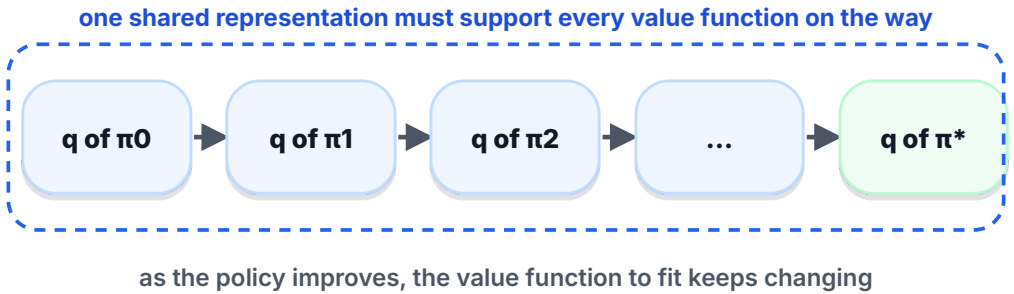
Capture the flag: find the other team's flag and bring it home; the reward is winning. Add "how far is our flag?" and "am I being chased?". I don't care much about either on its own, but the network learns a representation that serves all of them, and winning gets easier.



IN PLAIN WORDS

Not every task helps. Add a task called "have a picnic", and the network learns something about picnics. It won't help you capture the flag.

The value-improvement path



- Why regularise the representation? During learning we approximate many value functions, because we track a continuously improving policy.
- The representation must support all functions on the **value-improvement path**, from q_{π_0} to q_{π_*} (Dabney et al., 2020).
- Which GVFs regularise the representation best? Ones that resemble the value functions along that path.



IN PLAIN WORDS

We don't fully understand how neural networks learn representations. We understand fully connected networks better every year, and we're already at transformers.

Check yourself: Part 2

CHECK YOURSELF

4. Why might an auxiliary task help, even if we never use its predictions?

CHECK YOURSELF

5. Give an example of an auxiliary task that would probably not help a driving agent.

CHECK YOURSELF

6. What is the value-improvement path?



Applied Computational Mathematics Lab

PART 3

Trade-offs and PopArt

Keep loud tasks from drowning quiet ones.



Learning about many things: trade-offs

- We want to learn as much as possible about the world, with limited resources (capacity, computation, ...).
- Different tasks compete for these resources. How do we trade off competing needs?
- There is always a trade-off, even if you do nothing explicit: the size of each task's updates to the shared weights differs.
- It scales linearly with the frequency and size of each task's rewards. A loud task pulls the representation its way.

TRY IT

Two tasks share a layer. Task A's rewards are around 1, task B's around 1,000. With the same errors in relative terms, B's gradients are about 1,000 times larger: the shared layer mostly learns about B.



IN PLAIN WORDS

Even if you do nothing explicit about it, you've made a choice.

Adaptive target normalisation

- In supervised learning we know the dataset beforehand, and can normalise inputs and targets. Many deep models don't work well without this.
- In RL we don't have the full dataset, and the scale of the values we predict changes over time. Solution: normalise the updates adaptively (van Hasselt et al., 2016):

ADAPTIVE TARGET NORMALISATION

1. Observe a target, e.g. $T_{t+1} = R_{t+1} + \gamma \max_a q_\theta(S_{t+1}, a)$.
2. Update the statistics, step size η (e.g. 0.001): $\mu_{t+1} = \mu_t + \eta(T_{t+1} - \mu_t)$, $\nu_{t+1} = \nu_t + \eta(T_{t+1}^2 - \nu_t)$, $\sigma_{t+1} = \sqrt{\nu_{t+1} - \mu_{t+1}^2}$.
3. The network outputs a normalised $\tilde{q}_\theta(s, a)$; update $\Delta\theta_t \propto \left(\frac{T_{t+1} - \mu_{t+1}}{\sigma_{t+1}} - \tilde{q}_\theta(S_t, A_t)\right) \nabla_\theta \tilde{q}_\theta(S_t, A_t)$.
4. Recover the value, $q_\theta(s, a) = \sigma_t \tilde{q}_\theta(s, a) + \mu_t$, for bootstrapping and acting.



IN PLAIN WORDS

Every task's targets get the same scale, roughly mean 0 and standard deviation 1, however big its rewards are.

Preserving outputs

- Every update to the normalisation statistics changes **all** outputs, even for unrelated states. That seems bad.
- We can avoid it. Typically $\tilde{q}_{W,b,\theta}(s) = W\phi_\theta(s) + b$. When μ, σ change, rescale the last layer by the opposite amount:

$$W'_t = \frac{\sigma_t}{\sigma_{t+1}} W_t, \qquad b'_t = \frac{\sigma_t b_t + \mu_t - \mu_{t+1}}{\sigma_{t+1}} \quad \Longrightarrow \quad \sigma_{t+1}\, \tilde{q}_{W'_t,b'_t,\theta_t}(s) + \mu_{t+1} = \sigma_t\, \tilde{q}_{W_t,b_t,\theta_t}(s) + \mu_t$$

- Then update W'_t, b'_t and θ_t as normal (e.g. SGD).
- The name: **PopArt**, "preserving outputs precisely, while adaptively rescaling targets" (van Hasselt et al., 2016).



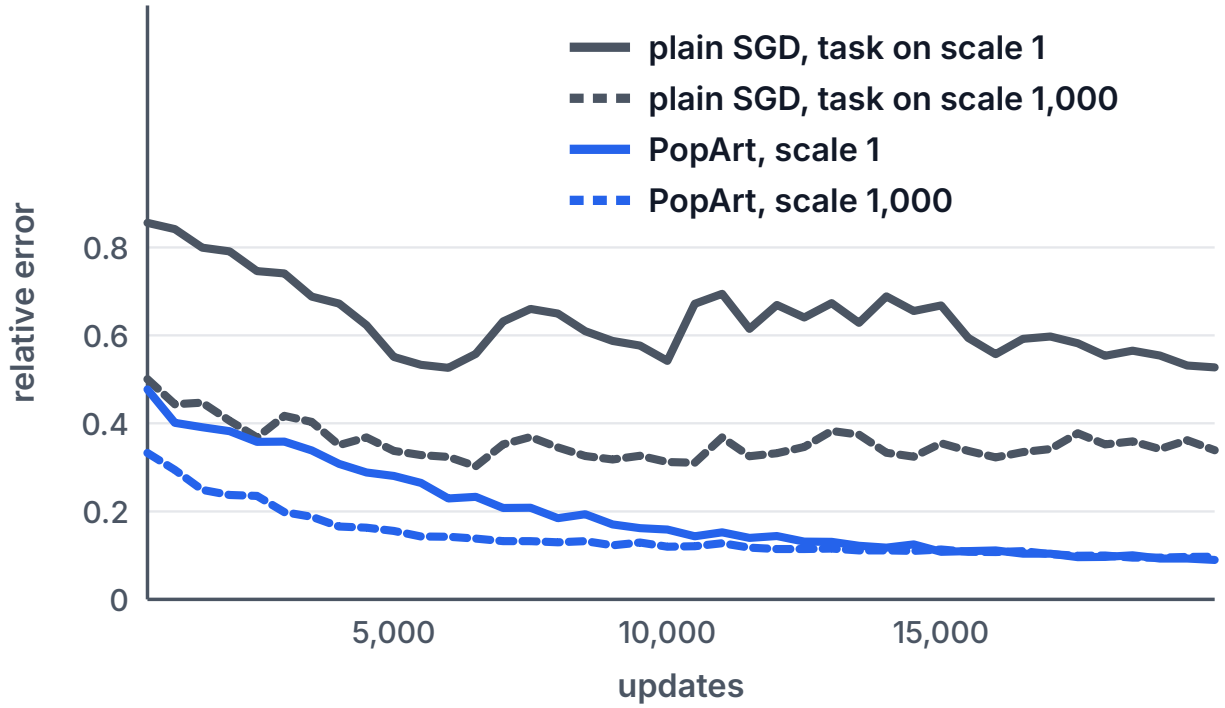
IN PLAIN WORDS

When the mean and spread move, rescale the last layer by exactly the opposite amount: the real outputs stay put.

TRY IT

$\sigma = 2$, $\mu = 1$, and a state's normalised output is 0.5: its value is $2 \times 0.5 + 1 = 2$. The statistics move to $\sigma' = 4$, $\mu' = 2$. With the rescaled W', b' the normalised output becomes $(2 \times 0.5 + 1 - 2)/4 = 0$, and $4 \times 0 + 2 = 2$: unchanged.

PopArt on a toy



Our run: two regression tasks share one hidden layer of 16 units; targets on scales 1 and 1,000; tasks alternate. Error relative to each task's scale.

METHOD (STEP SIZE)	SCALE 1	SCALE 1,000
plain SGD (0.001, best in our sweep)	0.53	0.34
plain SGD (0.0001)	0.83	0.10
PopArt (0.01)	**0.09**	**0.10**

- No single plain step size suits both tasks: small enough for the loud one is too slow for the quiet one. With PopArt both are learned well.

Check yourself: Part 3

CHECK YOURSELF

7. Why does a task with large rewards dominate a shared representation?

CHECK YOURSELF

8. $\mu = 0$, $\sigma = 1$; a target of 5 arrives. With $\eta = 0.1$ and $\nu = 1$, what are the new μ and ν?

CHECK YOURSELF

9. What does PopArt preserve when the statistics change, and why does that matter?



Applied Computational Mathematics Lab

PART 4

Open problems

Off-policy GVFs, universal value functions, and discovery.



Off-policy GVFs

- We can learn about one cumulant off-policy, using data from a policy that maximises a different cumulant.
- Learn about many things to learn to do the hard thing: “Unicorn” (Mankowitz et al., 2018) learned many tasks at once in one environment, sharing all the experience.
- Splitting a goal into sub-tasks is easy when you know the policy that collected the data; off-policy it is much less clear how, and the deadly triad is never far away.



IN PLAIN WORDS

One stream of experience, many lessons: the same idea as Lecture 6, now with many cumulants instead of one reward.

EXAMPLE

A household robot collects data while tidying. From the same data, it learns how to reach the kitchen, the door and the charger, each as a separate GVF with its own target policy.

Universal value functions

- GVF auxiliary tasks help each other by sharing representations; otherwise each prediction is learned independently.
- Can we generalise what we learn about one GVF to others?
- Idea: feed a representation of (c, γ), or of a goal g, as an **input** (Schaul et al., 2015):

$$v(s, g; \theta) \approx v_g(s)$$

- This allows generalisation across goals and tasks within one environment. These are **universal value function approximators** (UVFAs).

TRY IT

A navigation agent trained to reach rooms 1, 2 and 4, with the goal room as an input. Asked about room 3, which it never trained on, it can still make a sensible guess: goals are now points in a space it can generalise over.



IN PLAIN WORDS

A table of value functions has no neighbourhoods. A function of the goal does.

Discovering what to predict

- Which GVFs best regularise the representation?
- The value-improvement path gives one general answer.
- Alternatively, learn from experience which cumulants to predict, with a form of **meta-learning** (Veeriah et al., 2019).
- Meta-gradient RL provides a concrete mechanism: adjust the cumulants so that learning them improves the main task.



IN PLAIN WORDS

It's a field I love. My own little corner is meta-learning for feature selection. Everyone needs their own little corner.

IN PLAIN TERMS

Instead of a human deciding "predict the distance to the flag", the agent tries out candidate questions and keeps the ones whose answers make its main task easier.

Applied Computational Mathematics Lab

PART 5

Distributional RL

When one number is not enough.



Same mean, different worlds

- GVFs still represent knowledge as **expectations**.
- We could instead learn the **distribution of returns**, not just its expected value.
- This also gives a richer learning signal, which can help learn more robust representations.

BET	MEAN	WHAT ACTUALLY HAPPENS
Always 0	0	0, every time
A coin: +10 or −10	0	never 0
A die: 1 to 6	3.5	never 3.5



IN PLAIN WORDS

Same average, very different worlds. One of them you'd like to know about in advance.

THE CATCH

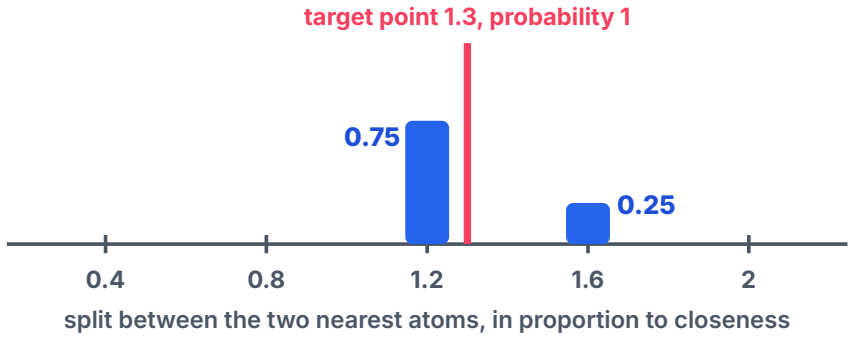
We have endless tools to predict a number. We're much less good at predicting distributions: two humps, a long tail. There are tools, but fewer, and they are harder to use.

Categorical return distributions

- Categorical DQN (Bellemare et al., 2017): a fixed "comb" of support points, e.g. 51 atoms $\mathbf{z}$ from −10 to 10 (hence "C51").
- For each atom, a probability $p_\theta^i(s,a)$. The return distribution of (s,a) is approximated by $(\mathbf{z}, \mathbf{p}_\theta(s,a))$.
- The expectation $\mathbf{z}^\top \mathbf{p}_\theta(s,a) \approx q(s,a)$ is used to act.

CATEGORICAL TD UPDATE

1. Find the max action: $a^* = \operatorname{argmax}_a \mathbf{z}^\top \mathbf{p}_\theta(S_{t+1}, a)$ (e.g. with θ^- for double Q).
2. Shift and shrink the support: $\mathbf{z}' = R_{t+1} + \gamma \mathbf{z}$.
3. Project $(\mathbf{z}', \mathbf{p}_\theta(S_{t+1}, a^*))$ back onto $\mathbf{z}$: $d' = (\mathbf{z}, \mathbf{p}')$.
4. Minimise $\mathrm{KL}(d' \| d) \to -\sum_i p'_i \log p_\theta^i(S_t, A_t)$.

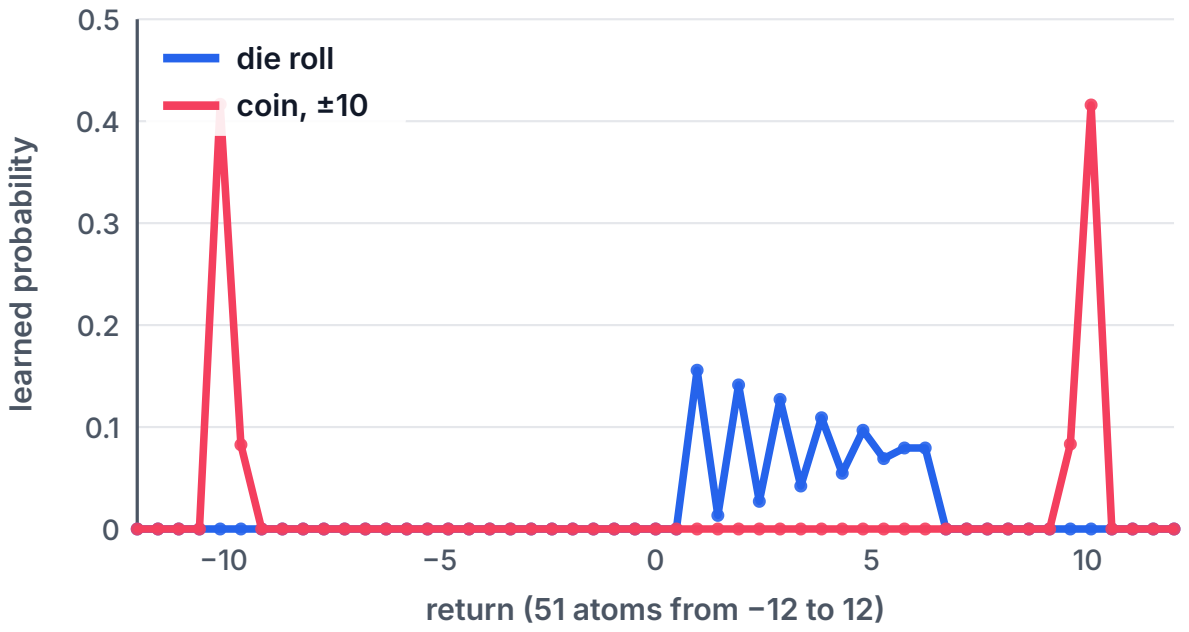


The projection step, with atoms 0.4 apart. Our drawing.

TRY IT

A shifted atom lands at 1.3, between atoms 1.2 and 1.6. It gives $(1.6 - 1.3)/0.4 = 0.75$ of its probability to 1.2 and 0.25 to 1.6.

Categorical TD on a die and a coin



Our run: one-step returns, categorical TD with 51 atoms, 200,000 samples each.

- Die: about one sixth of the probability near each face, 1 to 6 (spread over neighbouring atoms, which are 0.48 apart). Its mean: 3.45; the truth: 3.5.
- Coin: two peaks, at −10 and +10, and nothing in between. Mean -0.003; the truth: 0.



IN PLAIN WORDS

The mean alone would call the coin “worth 0” and stop there. The distribution says you’ll never actually get 0.

Quantile regression

- Many other ways to model return distributions exist. **Quantile regression** (Dabney et al., 2017) transposes the parametrisation:
- instead of adjusting the probabilities of a fixed support (C51),
- adjust the support (the locations) for a fixed set of probabilities.
- For quantile levels τ_i, each location moves up by τ_i when a sample lands above it and down by $1 - \tau_i$ when below:

$$q_i \leftarrow q_i + \alpha\big(\tau_i - \mathbb{I}(r < q_i)\big)$$

τ	0.04	0.23	0.41	0.59	0.77	0.95
die	1	2	2.99	3.97	5	6.02
coin	-10.02	-10	-9.99	9.99	10	10.02

Our run: 11 quantiles, 40,000 samples; every other quantile shown.



IN PLAIN WORDS

On the coin, five quantiles land at −10, five at +10, and the middle one wanders in between (2.86 in our run). With a coin, that's honest: any value between −10 and +10 is a median.

Check yourself: Parts 4 and 5

CHECK YOURSELF

10. What does a universal value function take as input that an ordinary GVF does not?

CHECK YOURSELF

11. In C51 with atoms 0.5 apart, a shifted atom lands at 2.1, between 2.0 and 2.5. How is its probability split?

CHECK YOURSELF

12. In quantile regression with $\tau = 0.9$, a sample lands below the current quantile. Which way does the quantile move, and by how much (step α)?



WRAP-UP

Summary

- Narrow objectives give narrow representations: learn about many things.
- **GVFs**: a cumulant, a discount and a target policy turn any signal into a value-function question.
- **Auxiliary tasks** shape a shared representation; the value-improvement path says what it must support.
- Tasks compete: loud rewards dominate. **PopArt** normalises targets adaptively and preserves outputs.
- **UVFAs** generalise across goals; meta-learning can **discover** what to predict.
- **Distributional RL** learns the whole return distribution: categorical (C51) or quantile.

$$q_{c,\gamma,\pi}(s,a) = \mathbb{E}\big[\textstyle\sum_k (\prod_{j\le k} \gamma_{t+j}) C_{t+k+1}\big]$$

$$q = \sigma\tilde{q} + \mu$$

$$\mathbf{z}' = R + \gamma\mathbf{z}$$

ANSWERS

Answers to check yourself

- **1.** Cumulant (what to add up), discount (horizon / termination), target policy (whose behaviour).
- **2.** Cumulant = reward, discount 0, the agent's policy.
- **3.** From cell 0 or cell 1, the next step reaches cell 0 with probability 1/2 (from cell 0 by bumping, from cell 1 by stepping left).
- **4.** It shapes the shared layers into a richer representation.
- **5.** Anything unrelated to driving, like predicting the colour of the sky.
- **6.** The sequence of value functions of the improving policies, from q_{π_0} to q_{π_*}.
- **7.** Its gradients scale with the size of its rewards.
- **8.** $\mu = 0.5$, $\nu = 1 + 0.1(25 - 1) = 3.4$ (so $\sigma = \sqrt{3.15} \approx 1.77$).
- **9.** The real (unnormalised) outputs of every state: changing the statistics shouldn't change the values of unrelated states.
- **10.** A representation of the goal (or of the cumulant and discount).
- **11.** $(2.5 - 2.1)/0.5 = 0.8$ to 2.0 and 0.2 to 2.5.
- **12.** Down, by $\alpha(1 - 0.9) = 0.1\alpha$.

CREDITS

Sources and credits

- The order of topics follows *Deep Reinforcement Learning, Part 2* of the DeepMind × UCL Reinforcement Learning Lecture Series (2021), by Matteo Hessel. 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 et al. (2011), GVFs; Littman et al. (2002), PSRs; Jaderberg et al. (2016), UNREAL; Dabney et al. (2020), the value-improvement path; van Hasselt et al. (2016), PopArt; Mankowitz et al. (2018), Unicorn; Schaul et al. (2015), UVFAs; Veeriah et al. (2019), discovery; Bellemare et al. (2017), C51; Dabney et al. (2017), quantile regression. The corridor, PopArt toy and die/coin runs are ours.
- Companion videos: **ACML Explains**, season 1, episodes 43–46. More from the lab: acml.teddylazebnik.com



Applied Computational Mathematics Lab

END OF LECTURE 11

Next: learning from people

Lecture 12, the last: imitation learning, inverse RL, and reinforcement learning from human feedback.

