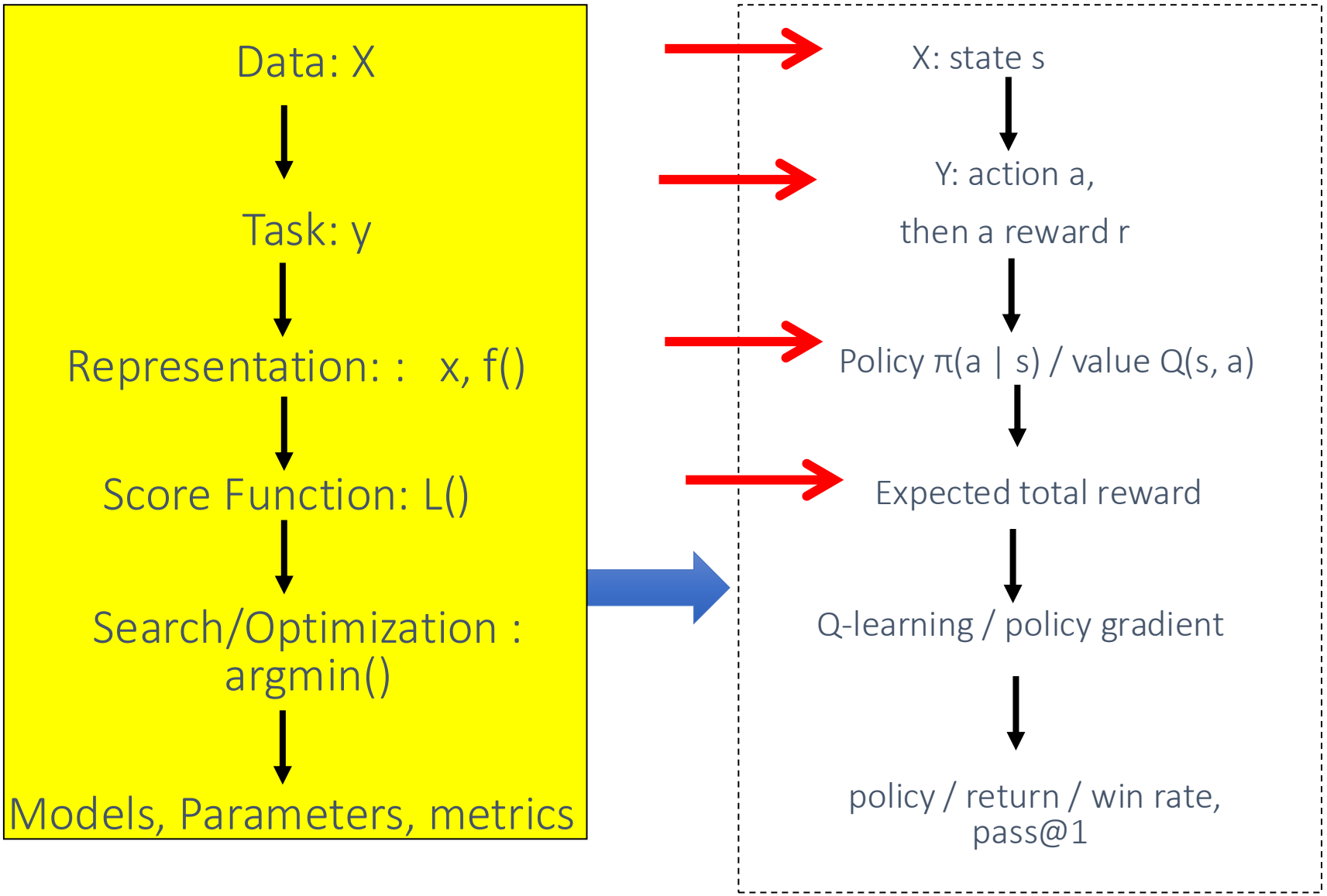
UVA CS 4774: Machine Learning

S3: Lecture 19b: Reinforcement Learning: from Q-learning to RLHF and GRPO

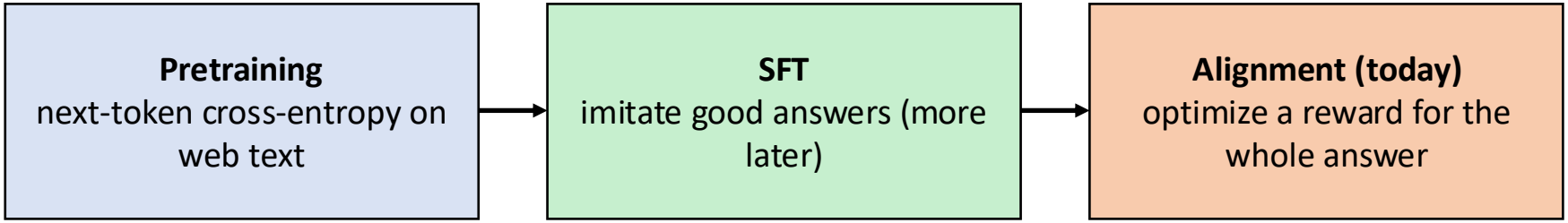
Dr. Yanjun Qi

University of Virginia
Department of Computer Science

Today: Reinforcement Learning → RLHF / GRPO



Recap: where we left off



- Cross-entropy rewards copying the data, token by token.
- What we want is a property of the whole answer: helpful, correct, safe.
- There is no single "right next token" for that, only a **score** after the answer is written.
- Learning from a score that comes after a sequence of choices = reinforcement learning.

Today: what is the "RL" in RLHF, and how did GRPO teach DeepSeek-R1 to reason?

Roadmap : Today's Lecture

- **What is reinforcement learning?**
- Value-based RL: V, Q, Bellman, Q-learning
- Policy gradients: REINFORCE, actor-critic, PPO
- RLHF and DPO: aligning an LLM
- RL with verifiable rewards: GRPO, DeepSeek-R1

https://36708.github.io/index.html

Classes of Learning Problems

Supervised Learning:

Data: (x, y)

x is data, y is label

Goal: Learn function to map x → y

Example:



This thing is an apple.

Unsupervised Learning:

Data: x

x is data, no labels!

Goal: Learn underlying structure

Example:



This thing is like the other thing.

Reinforcement Learning:

Data: state-action pairs

Goal: Maximize future rewards over many steps

Example:



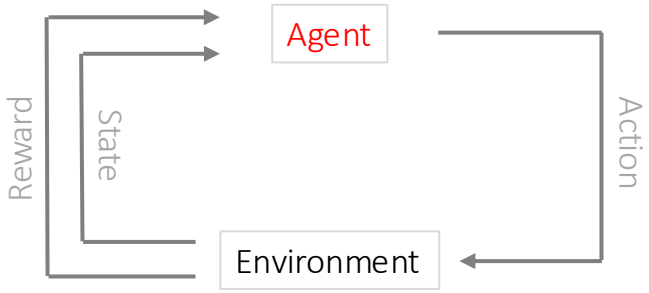
Eat this thing because it will keep you alive.

Reinforcement Learning

- Learning to interact with an environment
  - Robots, games, process control
  - With limited human training
  - Where the 'right thing' isn't obvious

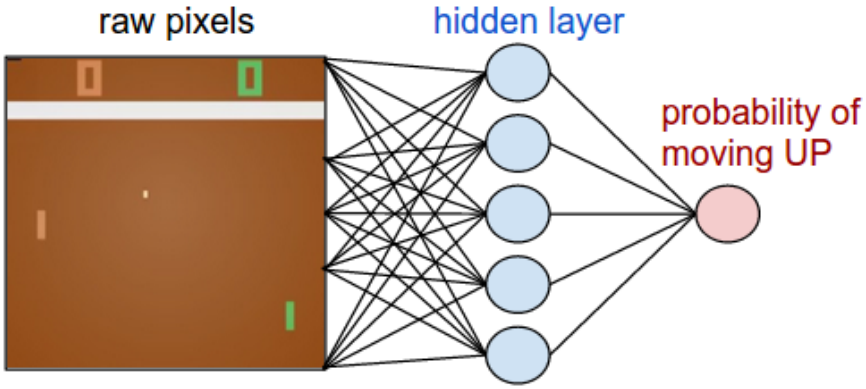
- Supervised Learning:
  - Goal: $f(x) = y$
  - Data: $[< x_1, y_1 >, \dots, < x_n, y_n >]$

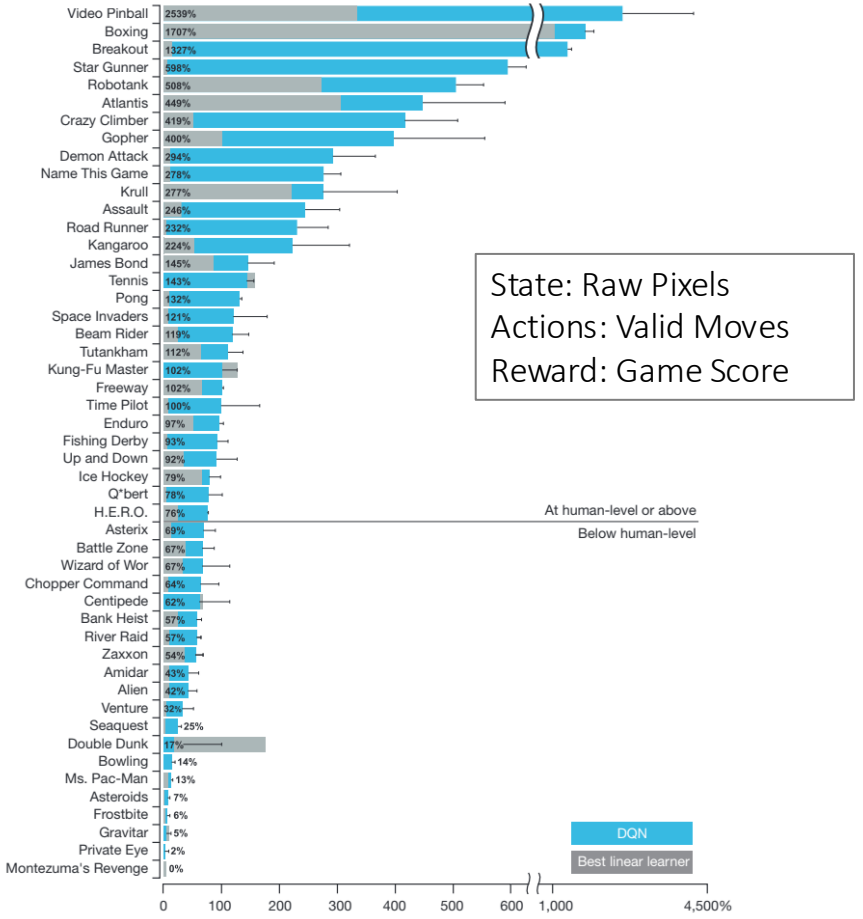
- Reinforcement Learning:
  - Goal:
    Maximize $\sum_{i=1}^{\infty} Reward(State_i, Action_i)$
  - Data:
    $Reward_i, State_{i+1} = Interact(State_i, Action_i)$



credit: G. Hulten (via the UVA CS 4774 RL deck)

Atari 2600 games





State: Raw Pixels
Actions: Valid Moves
Reward: Game Score

- Same model/parameters for ~50 games

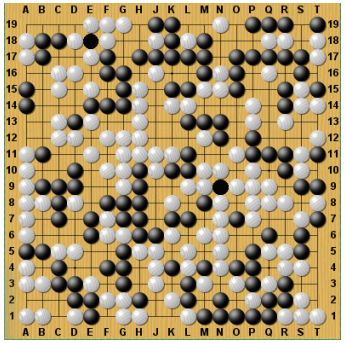
https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf

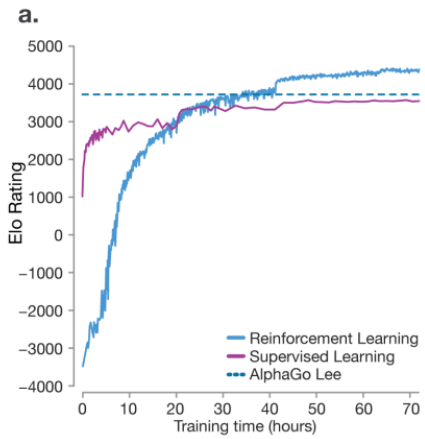
credit: G. Hulten (via the UVA CS 4774 RL deck); Mnih et al., Nature 2015

Alpha Go

State: Board State
Actions: Valid Moves
Reward: Win or Lose

- Learning how to beat humans at 'hard' games (search space too big)
- Far surpasses (Human) Supervised learning
- Successor AlphaZero learned to outplay humans at chess in 24 hours







https://deepmind.com/documents/119/agz_unformatted_nature.pdf

credit: G. Hulten (via the UVA CS 4774 RL deck); Silver et al., Nature 2017; AlphaZero: Silver et al., Science 2018

What is special about RL?

- RL learns to map states to actions so as to maximize a numerical reward over time.
- It is a multi-step decision-making process (usually Markovian).
- The agent learns by trial and error: not supervised, but interactive.
  - The agent chooses its own training data; exploration matters.
- Actions affect the immediate reward and all later rewards (delayed effect).
  - Which of my 100 moves (or 500 tokens) deserves the credit? → **credit assignment**

credit: E. Xing, CMU 10-701 (via the UVA CS 4774 RL deck)

Elements of RL

- **A policy** $\pi(a \mid s)$
  - a map from the state space to the action space; may be stochastic
- **A reward** r
  - maps each state (or state-action pair) to a real number
- **A value function** V(s) or Q(s, a)
  - the total expected reward starting from that state (or state-action pair)
- **(Optional) a model** of the environment: $P(s' \mid s, a)$
  - model-based RL plans with it; **model-free** RL (today) learns without it

reward = what is good **now**
value = what is good **in the long run**

credit: E. Xing, CMU 10-701 (via the UVA CS 4774 RL deck)

Setup: Markov decision process (MDP)

- States s, actions a, discrete time steps t = 0, 1, 2, …
- At step t: the agent is in state s_t, picks $a_t \sim \pi(a \mid s_t)$
  - the environment returns reward r_t and the next state $s_{t+1} \sim P(s' \mid s_t, a_t)$
- **Markov property:** the future depends only on s_t, not on the history
- Return from step t (discount $0 \le \gamma < 1$):

$$G_t = r_t + \gamma\, r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k \ge 0} \gamma^k\, r_{t+k}$$

$$V^\pi(s) = \mathbb{E}_\pi\left[\, G_t \mid s_t = s \,\right] \qquad Q^\pi(s,a) = \mathbb{E}_\pi\left[\, G_t \mid s_t = s,\ a_t = a \,\right]$$

Goal: find the policy π with the largest value in every state.

credit: G. Hulten (via the UVA CS 4774 RL deck); Sutton & Barto 2018

Simple Example of Agent in an Environment

State:

Map Locations

$\{< 0,0 >, < 1,0 > \cdots < 3,3 >\}$

Actions:

Move within map

Reaching chest ends episode

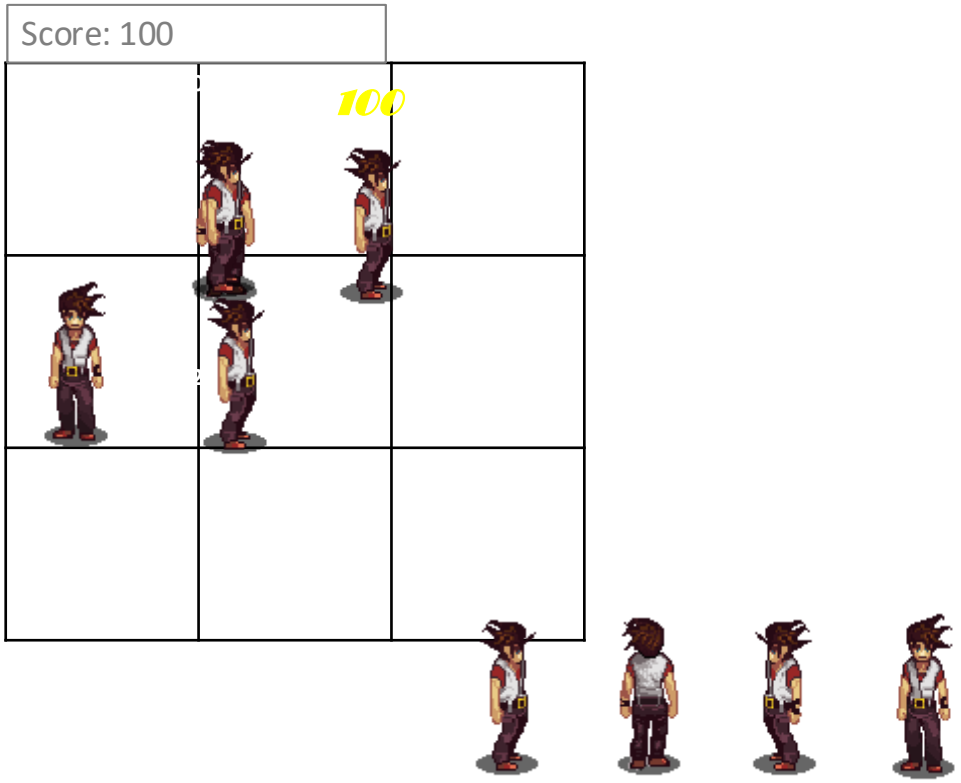
$A_{0,0} = \{ east, south \}$
$A_{1,0} = \{ east, south, west \}$
$A_{2,0} = \{ \phi \}$
...
$A_{2,2} = \{ north, west \}$

Reward:

100 at chest

0 for others

$R_{east}(< 1,0 >, < 2,0 >) = 100$
$R_{north}(< 2,1 >, < 2,0 >) = 100$
$R_*(*,*) = 0$



credit: G. Hulten (via the UVA CS 4774 RL deck)

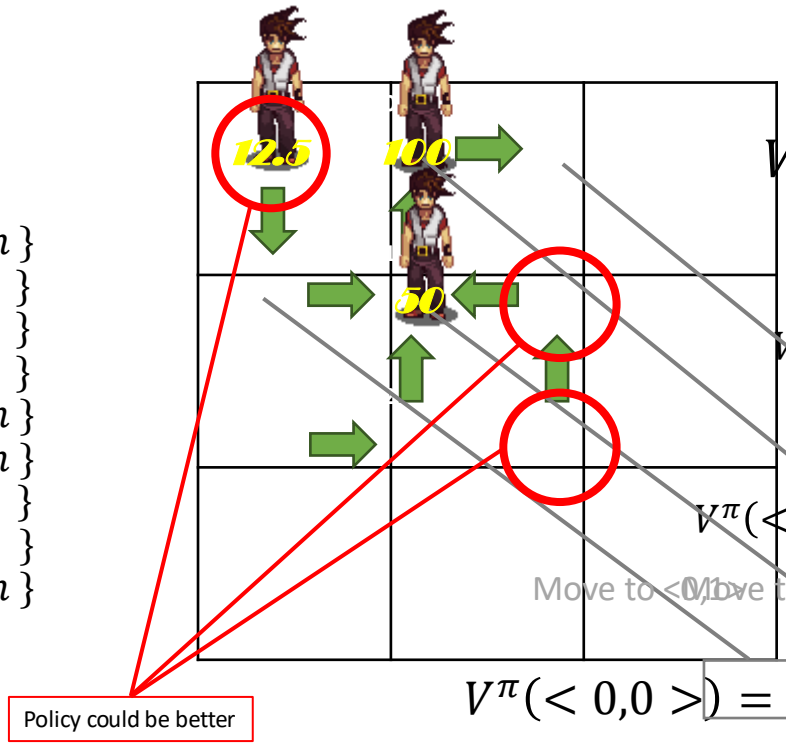
Policies

$R_{east}\ (<1,0>,<2,0>) = 100$
$R_{north}(<2,1>,<2,0>) = 100$
$R_{*}\ \ (*,*) = 0$
$\gamma = 0.5$

Policy

$$\pi(s) = a$$

$\pi(<0,0>) = \{south\}$
$\pi(<0,1>) = \{east\ \}$
$\pi(<0,2>) = \{east\ \}$
$\pi(<1,0>) = \{east\ \}$
$\pi(<1,1>) = \{north\}$
$\pi(<1,2>) = \{north\}$
$\pi(<2,0>) = \{\ \phi\ \}$
$\pi(<2,1>) = \{west\ \}$
$\pi(<2,2>) = \{north\}$



Evaluating Policies

$$V^{\pi}(s) = \sum_{i=0}^{\infty} \gamma^{i} r_{i+1}$$

$V^{\pi}(<1,0>) = \gamma^{0} * 100$

$V^{\pi}(<1,1>) = \gamma^{0} * 0 + \gamma^{1} * 100$

$V^{\pi}(<0,0>) = \gamma^{0} * 0 + \gamma^{1} * 0 + \gamma^{2} * 0 + \gamma$

credit: G. Hulten (via the UVA CS 4774 RL deck)

Exploration vs Exploitation

- Objective: max cash from slot machines as fast as possible

- Exploit: select only one slot machine and keep pulling the lever all day long
- Explore: Pull one lever of each and every machine at the casino
- Either approaches aren't optimal. Need to strike a balance



credit: UVA CS 4774 basic-RL deck (2023)

Roadmap : Today's Lecture

- What is reinforcement learning?
- **Value-based RL: V, Q, Bellman, Q-learning**
- Policy gradients: REINFORCE, actor-critic, PPO
- RLHF and DPO: aligning an LLM
- RL with verifiable rewards: GRPO, DeepSeek-R1

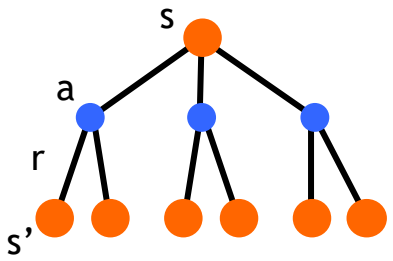
The goal and the classic algorithms

- Find a policy π that maximizes the value function.
  - transitions P(s' | s, a) and rewards are usually **not** known
- Dynamic programming (policy iteration, value iteration)
  - needs a full model of the environment
- Monte Carlo: learn from complete episodes
- Temporal-difference (TD) learning, e.g. **Q-learning**
  - learns from every single step, no model needed
- Policy gradients (Part 3): learn the policy directly

credit: E. Xing, CMU 10-701 (via the UVA CS 4774 RL deck)

Value functions

- state value function: $V^\pi(s)$
  - expected return when starting in *s* and following π
- state-action value function: Q-function: $Q^\pi(s,a)$
  - expected return when starting in *s*, performing *a,* and following π
- useful for finding the optimal policy
  - can estimate from experience
  - pick the best action using $Q^\pi(s,a)$



$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^\pi(s') \right] = \sum_a \pi(s,a) Q^\pi(s,a)$$

- Bellman equation

credit: P. Bodik, UC Berkeley (via the UVA CS 4774 RL deck)

Optimal value functions

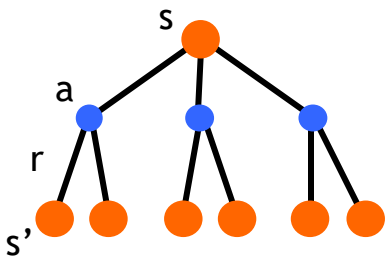
- there's a set of *optimal* policies
  - V^{π} defines partial ordering on policies
  - they share the same optimal value function

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

- Bellman optimality equation

$$V^*(s) = \max_{a} \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^*(s') \right]$$

  - system of n non-linear equations
  - solve for V*(s)
  - easy to extract the optimal policy



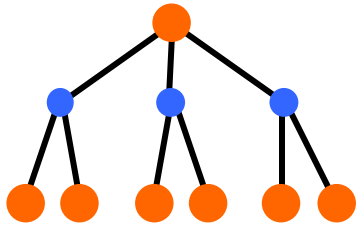
- having Q*(s,a) makes it even simpler

$$\pi^*(s) = \arg\max_{a} Q^*(s,a)$$

credit: P. Bodik, UC Berkeley (via the UVA CS 4774 RL deck)

Policy/Value iteration

- Policy iteration $\pi_0 \rightarrow^E V^{\pi_0} \rightarrow^I \pi_1 \rightarrow^E V^{\pi_1} \rightarrow^I \ldots \rightarrow^I \pi^* \rightarrow^E V^*$
  - two nested iterations; too slow
  - don't need to converge to V^{π_k}
    - just move towards it



- Value iteration $V_{k+1}(s) = \max_a \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V_k(s') \right]$
  - use Bellman optimality equation as an update
  - converges to V*

credit: P. Bodik, UC Berkeley (via the UVA CS 4774 RL deck)

Monte Carlo: learn from complete episodes

- No model needed: just experience (real or simulated)
- Play an episode with π; for every state visited, record the return G_t
- Estimate $V^{\pi}(s)$ = average of the returns observed from s
  - only defined for episodic tasks (the episode must end)
  - unbiased, but **high variance**: one lucky episode moves the average a lot
- Improve π greedily on the estimates; repeat

$$V(s_t) \leftarrow V(s_t) + \alpha \left[G_t - V(s_t) \right]$$

α = step size (as in gradient descent, earlier in this course). Not the attention weight αi,t of the attention lecture.

credit: P. Bodik, UC Berkeley (via the UVA CS 4774 RL deck)

Temporal Difference Learning

- combines ideas from MC and DP
  - like MC: learn directly from experience (don't need a model)
  - like DP: learn from values of successors
  - works for continuous tasks, usually faster than MC

- constant-alpha MC:
  - have to wait until the end of episode to update

$$V(s_t) \leftarrow V(s_t) + \alpha \left[G_t - V(s_t) \right]$$



target

- simplest TD
  - update after every step, based on the successor

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_t + \gamma V(s_{t+1}) - V(s_t) \right]$$



credit: P. Bodik, UC Berkeley (via the UVA CS 4774 RL deck)

Exploration: ε-greedy

- A purely greedy policy may never try the action that is actually best
  - at the start we know nothing about the environment
- **ε-greedy** policy:
  - with probability 1 − ε: take the greedy action $\text{argmax}_a Q(s, a)$
  - with probability ε: take a random action
- Keeps exploring; slowly decrease ε → 0 as Q becomes reliable
- LLM analogue: **sample** answers with temperature > 0 instead of greedy decoding

credit: P. Bodik, UC Berkeley (via the UVA CS 4774 RL deck)

Q learning

Learn a policy $\pi(s)$ that optimizes $V^{\pi}(s)$ for all states, using:

- No prior knowledge of state transition probabilities: $P_a(s, s')$
- No prior knowledge of the reward function: $R_a(s, s')$

Approach:

- Initialize estimate of discounted reward for every state/action pair: $\hat{Q}(s, a) = 0$
- Repeat (for a while):
  - Take a random action a from A_s
  - Receive s' and $R_a(s, s')$ from environment
  - Update $\hat{Q}(s, a) = R_a(s, s') + \gamma \max_{a'} \hat{Q}(s', a')$
  - Random restart if in terminal state

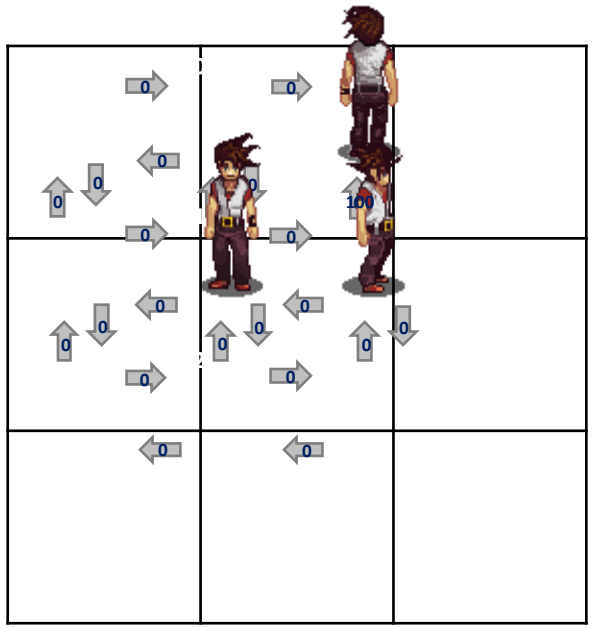
$$\propto_v = \frac{1}{1 + visits(s, a)}$$

Exploration Policy: $P(a_i, s) = \frac{k^{\hat{Q}(s,a_i)}}{\sum_j k^{\hat{Q}(s,a_j)}}$

credit: P. Bodik, UC Berkeley (via the UVA CS 4774 RL deck)

Example of Q learning (round 1)

- Initialize $\hat{Q}$ to 0
- Random initial state = $< 1,1 >$
- Random action from $A_{<1,1>} = east$
  - $s' =< 2,1 >$
  - $R_a(s, s') = 0$
- Update $\hat{Q}(< 1,1 >, east) = 0$
- Random action from $A_{<2,1>} = north$
  - $s' =< 2,0 >$
  - $R_a(s, s') = 100$
- Update $\hat{Q}(< 2,1 >, north) = 100$
- No more moves possible, start again…

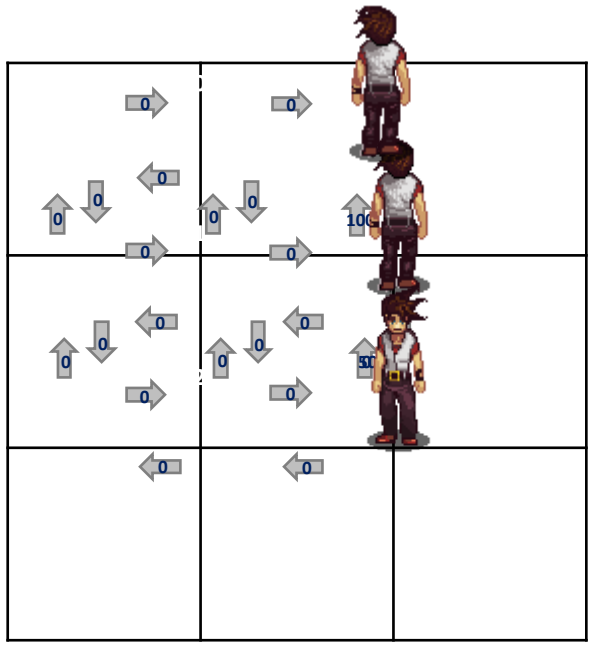


$$\hat{Q}(s,a) = R_a(s,s') + \gamma \max_{a'} \hat{Q}_{n-1}(s',a')$$

credit: P. Bodik, UC Berkeley (via the UVA CS 4774 RL deck)

Example of Q learning (round 2)

- Round 2: Random initial state = $< 2,2 >$
- Random action from $A_{<2,2>} = north$
  - $s' =< 2,1 >$
  - $R_a(s, s') = 0$
- Update $\hat{Q}(< 2,1 >, north) = 0 + \gamma * 100$



- Random action from $A_{<2,1>} = north$
  - $s' =< 2,0 >$
  - $R_a(s, s') = 100$
- Update $\hat{Q}(< 2,1 >, north) = still\ 100$

- No more moves possible, start again…

$$\hat{Q}(s,a) = R_a(s,s') + \gamma \max_{a'} \hat{Q}_{n-1}(s', a') = 0.5$$

credit: P. Bodik, UC Berkeley (via the UVA CS 4774 RL deck)

Quiz examples: one Q-learning update by hand

Same grid, γ = 0.5, deterministic moves. After rounds 1–2 the only non-zero entries are

$\hat{Q}$(<2,1>, north) = 100 and $\hat{Q}$(<2,2>, north) = 50

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$$

- Round 3 starts at <1,1> and moves **east** to <2,1> (reward 0). New $\hat{Q}$(<1,1>, east) = ?
- If it had moved **south** to <1,2> instead (reward 0), what would $\hat{Q}$(<1,1>, south) be?
- Why did <2,2> get a non-zero value only in round 2, not round 1?
- What does the final greedy policy do in <0,2> (bottom left)?

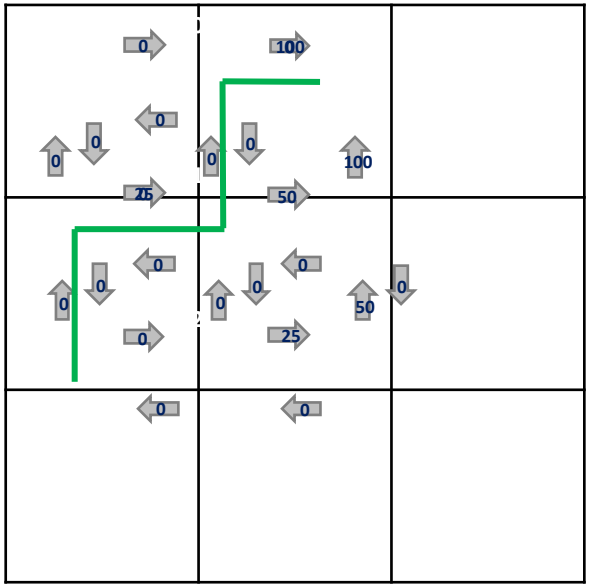
credit: example from P. Bodik, UC Berkeley (via the UVA CS 4774 RL deck)

Example of Q learning (some acceleration…)

$$\hat{Q}(s,a) = R_a(s,s') + \gamma \max_{a'} \hat{Q}_{n-1}($$

$$\gamma = 0.5$$

- Random Initial State $< 0,2 >$
- Update $\hat{Q}(< 0,1 >, east) = 25$
- Update $\hat{Q}(< 1,0 >, east) = 100$

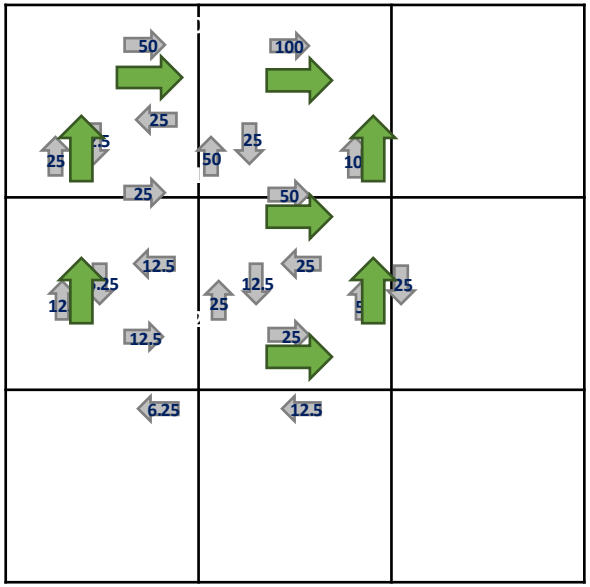


credit: P. Bodik, UC Berkeley (via the UVA CS 4774 RL deck)

Example of Q learning ($\hat{Q}$ after many, many runs…)

- $\hat{Q}$ converged
- Policy is:

$$\pi(s) = \underset{a \epsilon A_s}{\operatorname{argmax}} \hat{Q}(s, a)$$



credit: P. Bodik, UC Berkeley (via the UVA CS 4774 RL deck)

Deep Q-learning: a network instead of a table

- Atari has ~10^{100+} screens: no table can hold Q(s, a)
- Approximate Q(s, a; w) with a CNN (earlier: CNNs for images): pixels in, one Q-value per action out
- Train by regression toward the TD target:

$$\mathcal{L}(w) = \left(r + \gamma \max_{a'} Q(s', a'; w^{-}) - Q(s, a; w) \right)^2$$

- **Experience replay:** store transitions (s, a, r, s') in a buffer, train on random mini-batches
  - breaks the correlation between consecutive frames
- **Target network** w^{-}: a frozen copy of w, refreshed every C steps
  - the regression target does not move every step → stable training

credit: UVA CS 4774 basic-RL deck (2023); Mnih et al., Nature 2015

Roadmap : Today's Lecture

- What is reinforcement learning?
- Value-based RL: V, Q, Bellman, Q-learning
- **Policy gradients: REINFORCE, actor-critic, PPO**
- RLHF and DPO: aligning an LLM
- RL with verifiable rewards: GRPO, DeepSeek-R1

Policy-based vs value-based RL

Value-based (Part 2)

- learn Q(s, a); act by argmax
- no explicit policy
- argmax is hard with many / continuous actions

Policy-based (Part 3)

- learn $\pi_\theta(a \mid s)$ directly
- can learn stochastic policies
- works with huge action spaces (a vocabulary of 50k tokens!)
- but: local optima, high-variance gradient estimates

An LLM is already a **stochastic policy over tokens**, so LLM training uses policy gradients.

credit: UVA CS 4774 basic-RL deck (2023)

Policy gradient in one line

Objective: expected total reward of trajectories $\tau = (s_0, a_0, r_0, s_1, \ldots)$ sampled from π_θ

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\big[R(\tau)\big]$$

Log-derivative trick: $\nabla p = p \nabla \log p \Rightarrow$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\Big[R(\tau) \sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\Big]$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

- Sample, score, then raise the log-probability of what you did, in proportion to the reward
- Only needs samples and their rewards: R may be a game score, a human rating, or a unit test
- Replace samples with data tokens and set R = 1: this is the **next-token cross-entropy gradient**

credit: after Stanford CS224N W26 L08 (D. Yang); Williams 1992

REINFORCE: Monte Carlo policy gradient

- Sample complete episodes with π_θ; use the observed return G_t as the weight
  - unbiased, but high variance
- Subtract a baseline $b(s_t)$ from the return: same expected gradient, lower variance
- Better question: how much better than **average** was this action? → the advantage

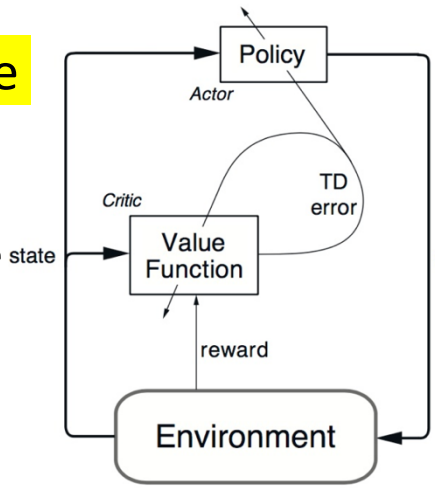
$$\nabla_\theta J \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t} \left(G_t - b(s_t) \right) \nabla_\theta \log \pi_\theta(a_t \mid s_t)$$

$$A(s,a) = Q(s,a) - V(s)$$

credit: UVA CS 4774 basic-RL deck (2023); Williams 1992

Actor Critic

- Instead of waiting until the end of an episode as in REINFORCE we use bootstrapping and make an update at each step.
- We train Critic that approximates the value function. Now we have two function approximators: One of the policy, one for the critic.
- **Critic**: updates value function parameters w and depending on the algorithm it could be action-value Q(a|s;w) or state-value V(s;w)
- **Actor**: updates policy parameters θ, in the direction suggested by the critic, π(a|s;θ)π(a|s;θ).



Actor-critic Architecture (https://arxiv.org/pdf/1509.0297

credit: UVA CS 4774 basic-RL deck (2023); Lillicrap et al. 2015

What multiplies ∇ log π? One table

All of today's methods use the same gradient; they differ only in the weight W_t:

$$\nabla_\theta J \approx \mathbb{E}\Big[\sum_t W_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\Big]$$

Method	**Weight W_t**	**Extra model?**
REINFORCE	return of the whole episode R(τ) or G_t	no
+ baseline	$G_t - b(s_t)$	maybe
Actor-critic, PPO	advantage $\hat{A}_t \approx Q(s_t, a_t) - V(s_t)$	critic V (a 2nd network)
GRPO (Part 5)	(r − mean) / std over G answers to the same prompt	no
SFT / pretraining	1 for every token of the data	no (and no sampling)

credit: adapted from CMU 11-711 ANLP S25 L23 (S. Welleck), "weighting variants"

PPO: Proximal Policy Optimization

- Policy gradient takes one step per batch of samples, and a too-large step can wreck the policy
- PPO reuses each batch for several steps, but clips how far the policy may move:

$$\rho_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

papers write $r_t(\theta)$ for the ratio; we use ρ_t because r is the reward

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\left(\rho_t(\theta)\,\hat{A}_t,\ \text{clip}(\rho_t(\theta), 1-\varepsilon, 1+\varepsilon)\,\hat{A}_t\right)\Big]$$

- $\hat{A} > 0$: raise the action's probability, but by at most a factor $1 + \varepsilon$
- $\hat{A} < 0$: lower it, but by at most $1 - \varepsilon$ ($\varepsilon \approx 0.2$; **not** the ε of ε-greedy)
- Stability of trust-region methods (TRPO) with a few lines of code; the RL step of InstructGPT / ChatGPT

credit: UVA CS 4774 basic-RL deck (2023); Schulman et al. 2017

Quiz examples: does a baseline bias the gradient?

For a fixed state s and any baseline b(s) that does not depend on the action a:

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[\, b(s)\,\nabla_\theta \log \pi_\theta(a \mid s)\,\right] \;= \; ?$$

- Hint: write the expectation as a sum over a, and use ∇log p = ∇p / p.
- What happens if the baseline depends on the action a?
- A group of 4 answers gets rewards 1, 1, 1, 0. With baseline = their mean, what weight does each answer get?

Roadmap : Today's Lecture

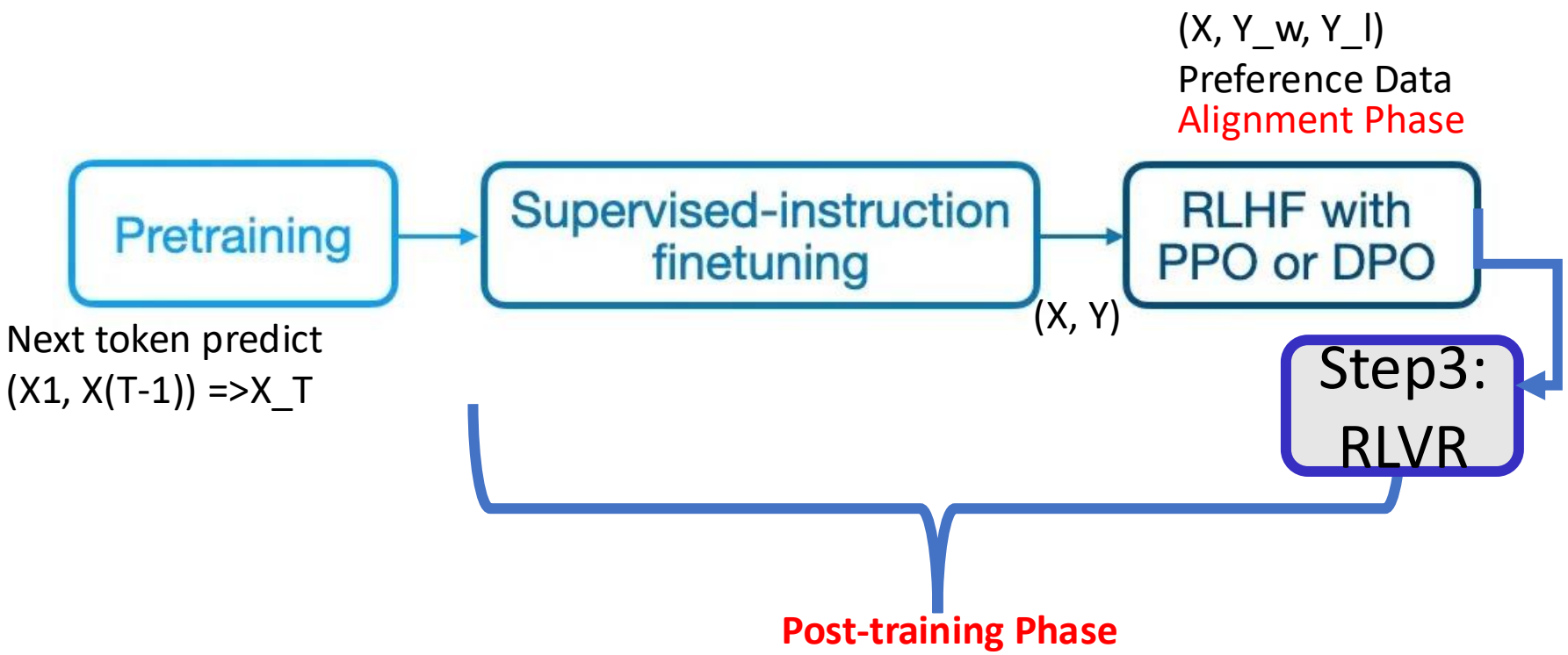
- What is reinforcement learning?
- Value-based RL: V, Q, Bellman, Q-learning
- Policy gradients: REINFORCE, actor-critic, PPO
- **RLHF and DPO: aligning an LLM**
- RL with verifiable rewards: GRPO, DeepSeek-R1

An LLM is a policy: the dictionary

RL	**LLM**	
state s_t	prompt x + tokens so far $y_{<t}$	same t as the output position in seq2seq and attention
action a_t	next token y_t	~50k possible actions
policy $\pi_\theta(a_t \mid s_t)$	$\pi_\theta(y_t \mid x, y_{<t})$	softmax at the last position of your GPT
transition	append y_t	deterministic
episode	one full answer $y = (y_1, \ldots, y_T)$	stops at <eos> or max length
reward	r(x, y), once, at the end	human-preference model or a checker

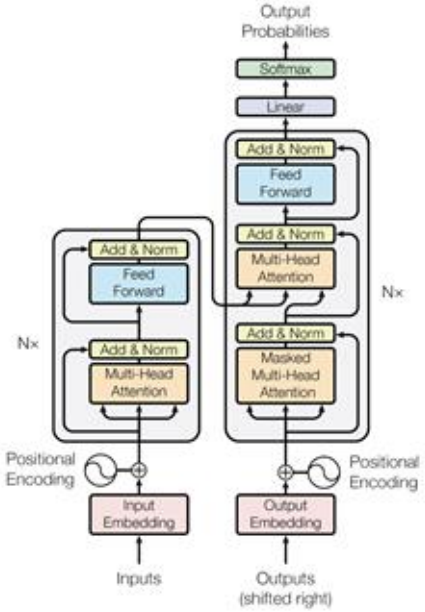
No new architecture. RL changes only the training signal: from cross-entropy on data to reward-weighted log-likelihood on the model's own answers.

Now: Post Training into Three Steps



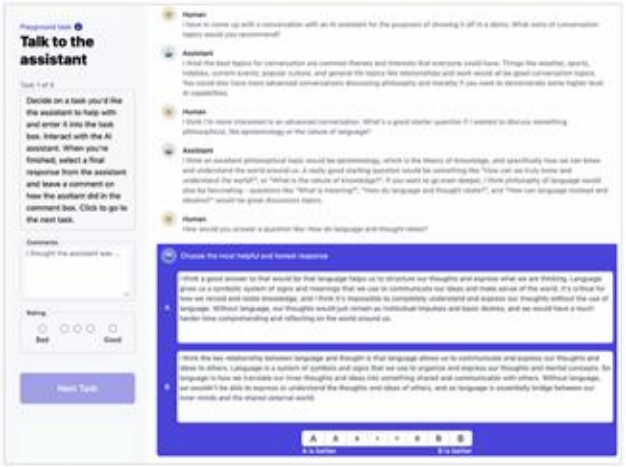
Build a Large Language Model (From Scratch) / Book by Sebastian Raschka

Three phases of RLHF

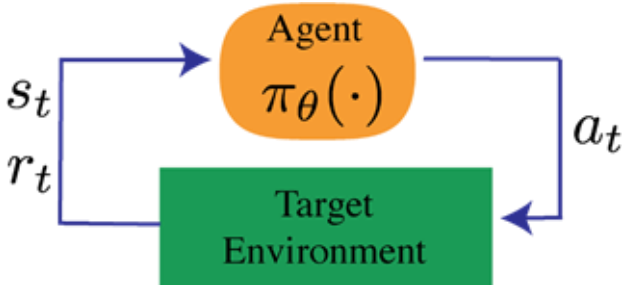


Vaswani et al. 2017

1. base model (instruction, helpful, chatty etc.)



2. preference collection & reward model training



3. reinforcement learning [policy optimization

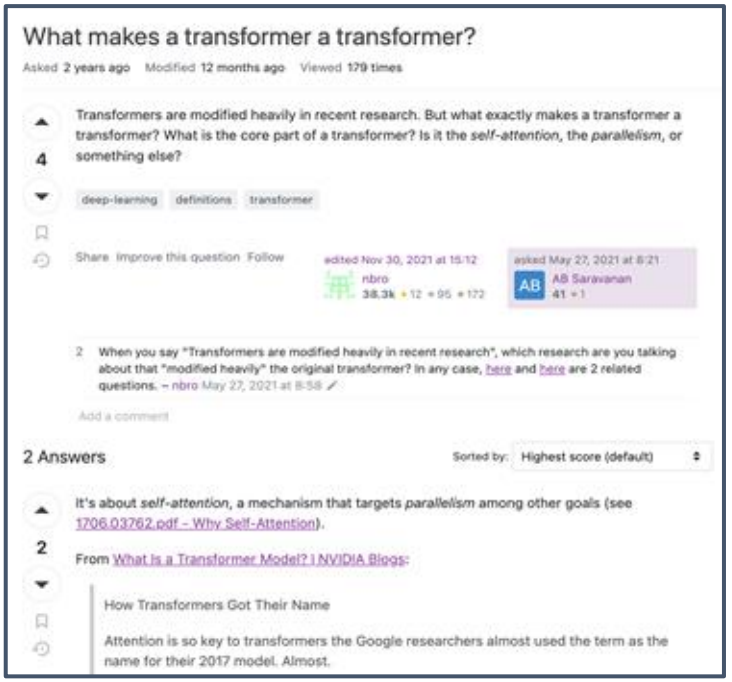
credit: N. Lambert et al., RLHF tutorial, ICML 2023

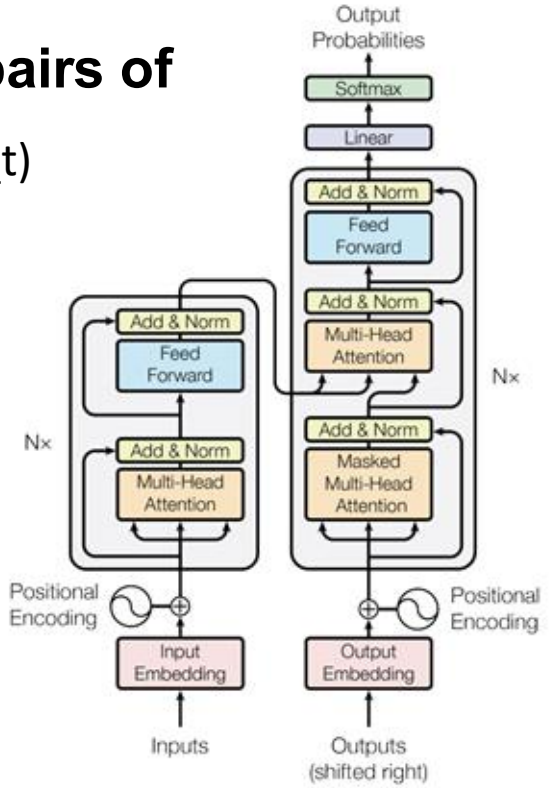
Instruction-tuned language model

starting point: a base language model

continue training a transformer with pairs of

question: answer fine training with (x_t, y_t)





Stack Overflow :*What makes a transformer a transformer?*, nbro 2021

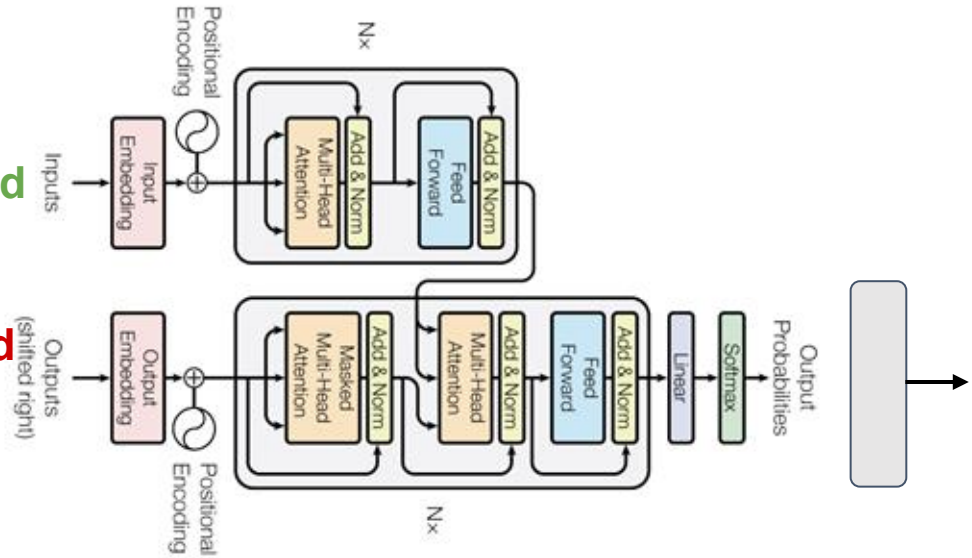
credit: N. Lambert et al., RLHF tutorial, ICML 2023

Reward /Preference model structure

input pair:

prompt + selected completion

prompt + rejected completion



The Transformer - Vaswani et al. 2017

outputs:
two scalar rewards

loss: increase difference of predicted reward

$$L_{\text{PM}} = \log\left(1 + e^{r_{\text{rejected}} - r_{\text{chosen}}}\right)$$

credit: N. Lambert et al., RLHF tutorial, ICML 2023

2. Modeling reward and training reward model:

Q: Human-in-the-loop is expensive!

Solution: Instead of asking humans directly, we train a separate reward model to learn human preferences.

$$L_{\mathrm{RM}}(r_\phi) = -\frac{1}{C_K^2}\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\phi(x,y_w) - r_\phi(x,y_l)\right)\right)\right]$$

y_w: winning sample

y_l: losing sample

y_w should score higher than y_l

credit: N. Lambert et al., RLHF tutorial, ICML 2023

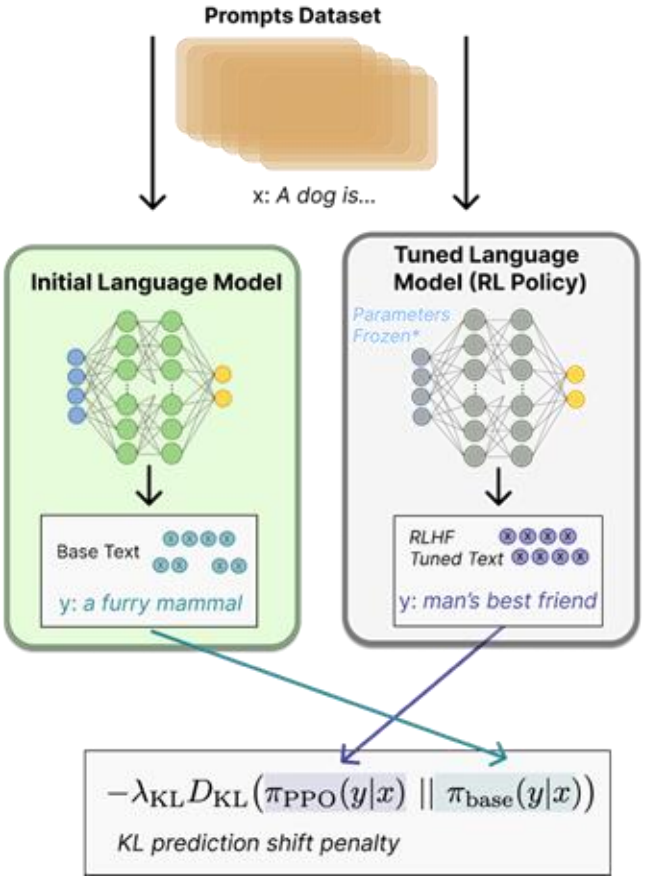
Fine tuning with RL – 3.2 KL penalty

Kullback–Leibler (KL) divergence: $D_{\mathrm{KL}}(P \parallel Q)$

How far p is from q: 0 if equal, not symmetric

Constrains the RL fine-tuning to not result in a LM that outputs gibberish (to fool the reward model).

Note: DeepMind did this in RL Loss (not reward), see GopherCite



credit: N. Lambert et al., RLHF tutorial, ICML 2023

Summary of Training Policy in RLHF

We have the following:

- A pretrained (possibly instruction-finetuned) LM $\pi_{ref}(y|x)$
- A reward model $r_\phi(x, y)$ that produces scalar rewards for LM outputs, trained on a dataset of human comparisons

Now to do RLHF:

$$\pi_\theta^*(y|x) = \max_{\pi_\theta} \mathbb{E}_{x\sim D}\left[\mathbb{E}_{y\sim\pi_\theta(y|x)} r_\phi(x,y) - \beta D_{\mathrm{KL}}(\pi_\theta(y|x)||\pi_{\mathrm{ref}}(y|x))\right]$$

Maximizing rewards

Minimizing divergence between current policy and reference policy

credit: N. Lambert et al., RLHF tutorial, ICML 2023

Results

InstructGPT: scaling up RLHF to 30k tasks!

Prompt

Explain the moon landing to a 6 year old in a few sentences.

Completion

GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

InstructGPT

```
People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them.
```


credit: Ouyang et al. 2022 (InstructGPT)

RLHF in practice

- Extract understanding from reward model (easy to overfit imperfect models)
- Memory and compute intensive (more gradients, runs can take days)
- Numerical instabilities during setup
  - Quantization
  - Loss regularization
  - Parallelization

credit: N. Lambert et al., RLHF tutorial, ICML 2023

(2) Can we simplify RLHF? Towards DPO

Direct Preference Optimization (DPO): directly optimizes policy based on human preference data using a clever loss function.

Recall our objective in RLHF:

$$\pi_\theta^*(y|x) = \max_{\pi_\theta} \mathbb{E}_{x\sim D}\left[\mathbb{E}_{y\sim\pi_\theta(y|x)} r_\phi(x,y) - \beta D_{\mathrm{KL}}(\pi_\theta(y|x)||\pi_{\mathrm{ref}}(y|x))\right]$$

There is a closed form solution to this:

$$\pi_\theta(y|x) = \frac{1}{Z(x)}\pi_{\mathrm{ref}}(y|x)e^{\left(\frac{1}{\beta}r_\theta(x,y)\right)}$$

Rearrange the terms:

$$r_\theta(x,y) = \beta\log\left(\frac{\pi_\theta(y|x)}{\pi_{\mathrm{ref}}(y|x)}\right) + \beta\log Z(x)$$

Reward model can be written in terms of policy!

credit: Rafailov et al. 2023 (DPO)

Can we simplify RLHF? Towards DPO

Direct Preference Optimization (DPO): directly optimizes policy based on human preference data using a clever loss function.

Recall, how we fit the reward model in RLHF:

$$L_{\mathrm{RM}}(r_\phi) = -\frac{1}{C_K^2}\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\phi(x,y_w) - r_\phi(x,y_l)\right)\right)\right]$$

Notice that we only need the difference between the rewards. Simplify for rewards:

$$r_\theta(x,y_w) - r_\theta(x,y_l) = \beta\left[\log\left(\frac{\pi_\theta(y_w|x)}{\pi_{\mathrm{ref}}(y_w|x)}\right) - \log\left(\frac{\pi_\theta(y_l|x)}{\pi_{\mathrm{ref}}(y_l|x)}\right)\right]$$

The final DPO loss function is:

$$-\mathbb{E}_{(x,y_w,y_l)\sim D}\log\left\{\sigma\left[\beta\log\left(\frac{\pi_\theta(y_w|x)}{\pi_{\mathrm{ref}}(y_w|x)}\right) - \beta\log\left(\frac{\pi_\theta(y_l|x)}{\pi_{\mathrm{ref}}(y_l|x)}\right)\right]\right\}$$

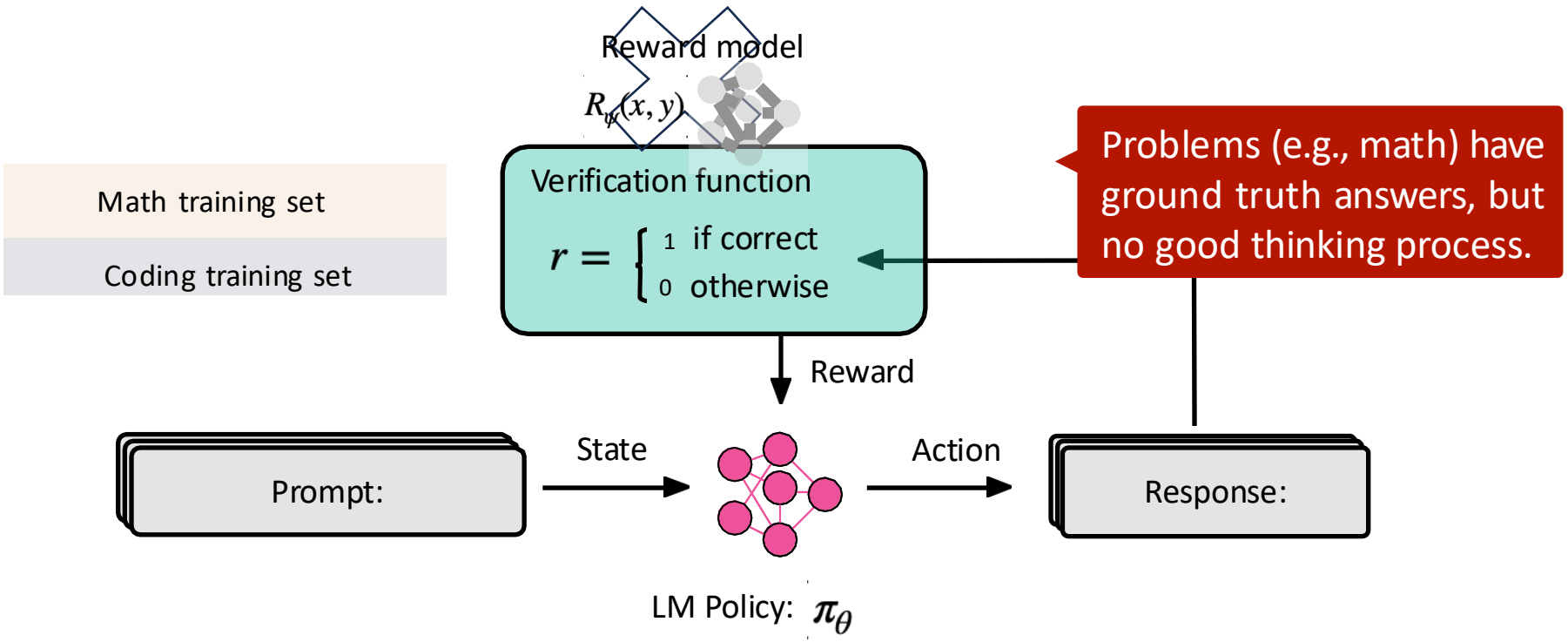
We have a classification loss function that connects **preference data** to **LM parameters** directly!

credit: Rafailov et al. 2023 (DPO)

Roadmap : Today's Lecture

- What is reinforcement learning?
- Value-based RL: V, Q, Bellman, Q-learning
- Policy gradients: REINFORCE, actor-critic, PPO
- RLHF and DPO: aligning an LLM
- **RL with verifiable rewards: GRPO, DeepSeek-R1**

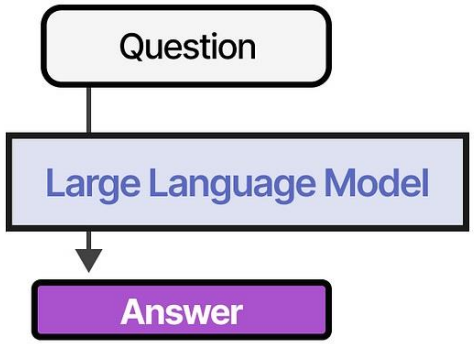
RL with verifiable rewards



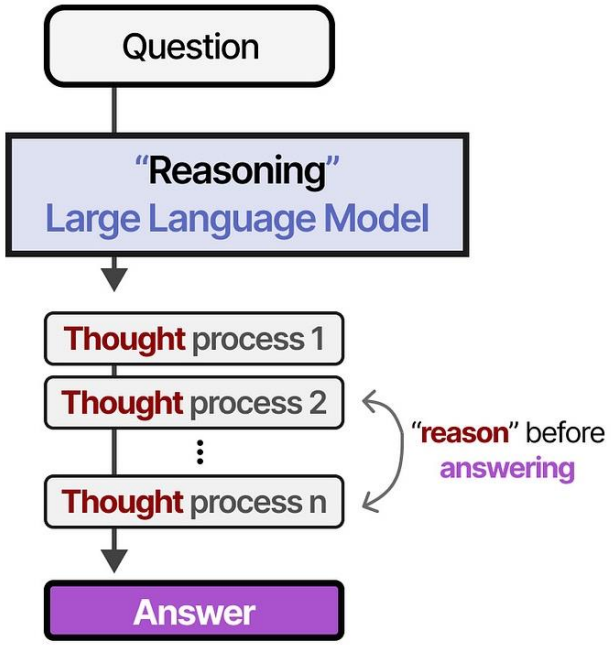
credit: N. Lambert et al., Tülu 3 (Ai2, 2024)

DeepSeek-R1: Using pure RL to effectively elicit reasoning behavior

"Regular" LLMs



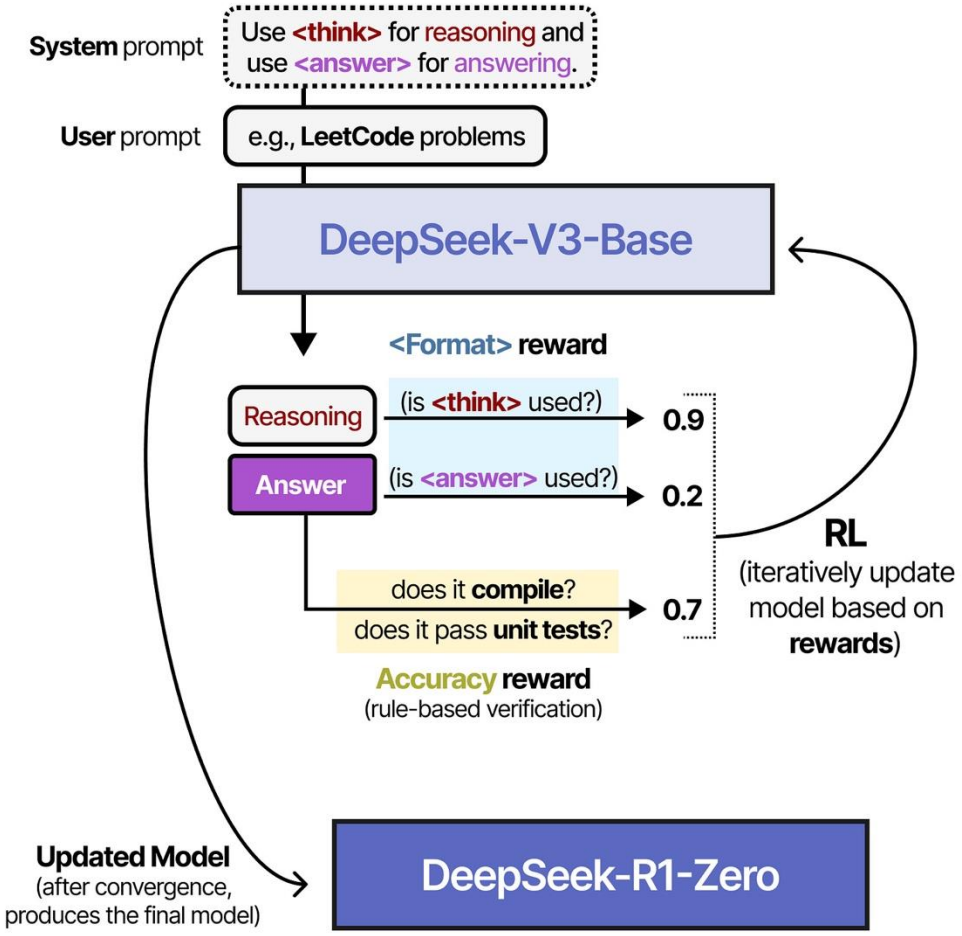
DeepSeek-R1 ("Reasoning" LLMs)



[Image credit: A Visual Guide to Reasoning LLMs]

credit: M. Grootendorst, A Visual Guide to Reasoning LLMs (2025); DeepSeek-AI 2025

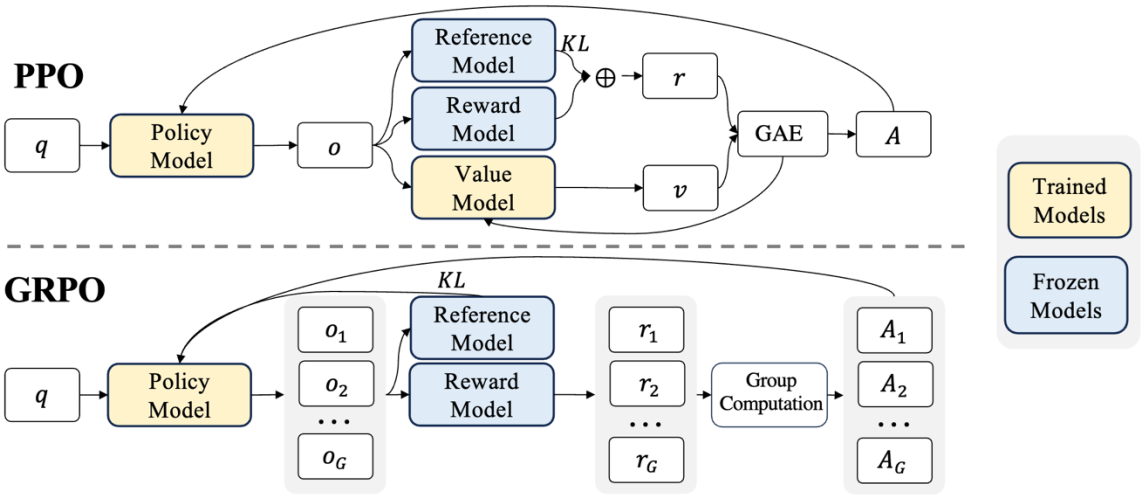
Reasoning with DeepSeek-R1 Zero



[Image credit: A Visual Guide to Reasoning LLMs]

credit: M. Grootendorst, A Visual Guide to Reasoning LLMs (2025); DeepSeek-AI 2025

Reasoning with DeepSeek-R1 Zero



$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right),$$

$$\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1,$$

Removal of the Value Model:

- PPO require a value model, while **GRPO remove the value model**.

Individual vs. Group:

- PPO uses a Value Model and Generalized Advantage Estimation to compute a single advantage.
- **GRPO** uses Group Computation to **calculate** relative **advantages for a group of outputs**.

Efficiency:

- **GRPO** avoids training and running an additional large Value Model, it is much more **efficient for scaling up** the training of LLMs.

credit: Shao et al. 2024 (DeepSeekMath); M. Grootendorst, A Visual Guide to Reasoning LLMs (2025); DeepSeek-AI 2025

GRPO in our notation

- For each prompt x, sample a group of G answers $y^{(1)}, \ldots, y^{(G)}$ from $\pi_{\theta old}$
- Score each one: $r^{(g)} = r(x, y^{(g)})$ (1 if correct, else 0; + format reward)
- Advantage = how much better than the group: the group mean is the baseline (no critic)

$$\hat{A}^{(g)} = \frac{r^{(g)} - \text{mean}\left(r^{(1)}, \ldots, r^{(G)}\right)}{\text{std}\left(r^{(1)}, \ldots, r^{(G)}\right) + \delta}$$

G = 4, rewards 1, 0, 0, 1:
mean 0.5, std 0.5 (÷G) → Â = +1, −1, −1, +1

$$J(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{g=1}^{G}\frac{1}{|y^{(g)}|}\sum_{t}\min\left(\rho_t^{(g)}\hat{A}^{(g)},\ \text{clip}(\rho_t^{(g)}, 1-\varepsilon, 1+\varepsilon)\,\hat{A}^{(g)}\right)\right] - \beta\, D_{\text{KL}}\left(\pi_\theta \,\|\, \pi_{\text{ref}}\right)$$

$$\rho_t^{(g)} = \frac{\pi_\theta(y_t^{(g)} \mid x, y_{<t}^{(g)})}{\pi_{\theta_{\text{old}}}(y_t^{(g)} \mid x, y_{<t}^{(g)})}$$

= PPO-clip (Part 3) + KL to π_{ref} (Part 4); all tokens of answer g share the advantage

Paper: q = prompt, o_i = answers; we keep x, y (q_t is the attention query vector)

credit: Shao et al. 2024 (DeepSeekMath); DeepSeek-AI 2025 (R1)

An interesting “aha moment” of DeepSeek-R1 Zero

Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a + x}} = x$ is equal to

Response: `<think>`

To solve the equation $\sqrt{a - \sqrt{a + x}} = x$, let's start by squaring both $\cdots$

$\left(\sqrt{a - \sqrt{a + x}}\right)^2 = x^2 \implies a - \sqrt{a + x} = x^2.$

Rearrange to isolate the inner square root term:

$(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$

$\cdots$

Wait, wait. Wait. That's an aha moment I can flag here.

Let's reevaluate this step-by-step to identify if the correct sum can be $\cdots$

We started with the equation:

$\sqrt{a - \sqrt{a + x}} = x$

First, let's square both sides:

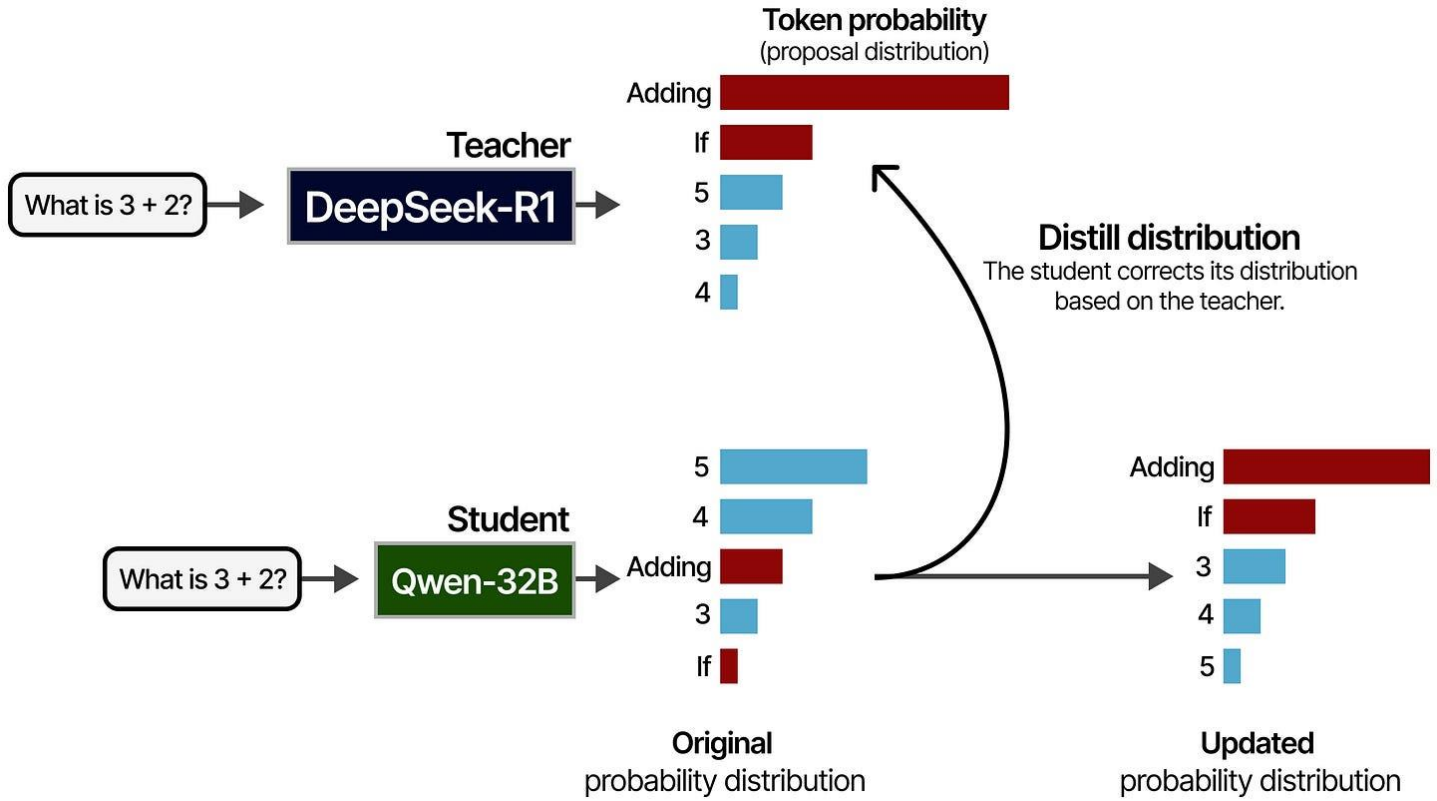
$a - \sqrt{a + x} = x^2 \implies \sqrt{a + x} = a - x^2$

Next, I could square both sides again, treating the equation: $\cdots$

$\cdots$

credit: M. Grootendorst, A Visual Guide to Reasoning LLMs (2025); DeepSeek-AI 2025; DeepSeek-R1 paper, Table 3

Distilling reasoning with DeepSeek-R1



DeepSeek-R1 serve as teacher model to distill reasoning to smaller student model.

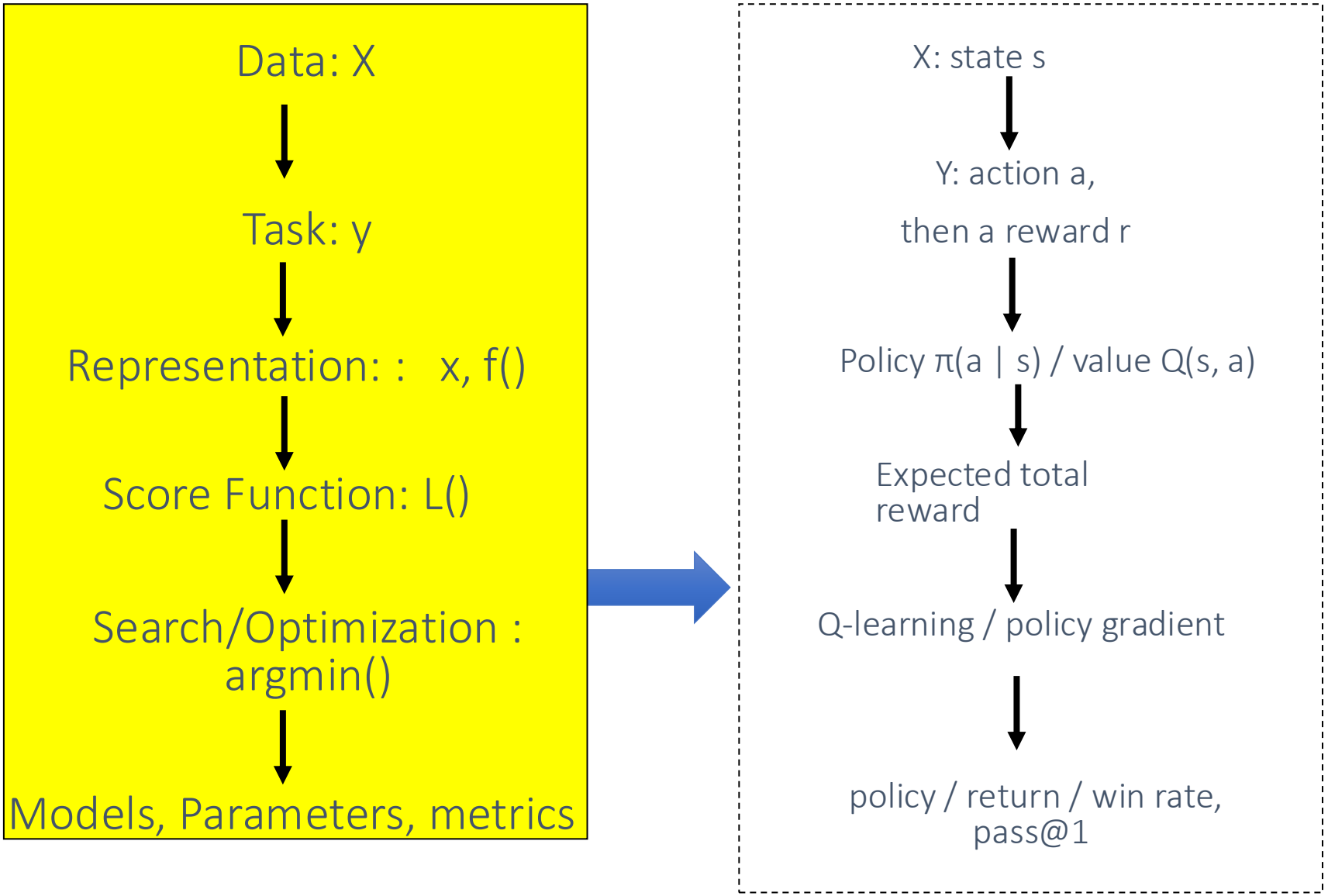
[Image credit: A Visual Guide to Reasoning LLMs]

credit: M. Grootendorst, A Visual Guide to Reasoning LLMs (2025); DeepSeek-AI 2025

Takeaways

- RL learns from a reward after a sequence of actions; value-based (Q-learning) or policy-based
- Policy gradient = $\nabla \log \pi$ of your own samples, weighted: return, advantage, or group-normalized reward
- **RLHF** = SFT → reward model from preference pairs → PPO with a KL penalty to π_{ref}
- **DPO** = the same objective as a classification loss on (y_w, y_l): no reward model, no RL loop
- **GRPO** = PPO without a critic: the group mean is the baseline; with verifiable rewards it taught DeepSeek-R1 to reason

Today Recap: Reinforcement Learning → RLHF / GRPO



References

- R. Sutton and A. Barto, Reinforcement Learning: An Introduction, 2nd ed., MIT Press 2018
- Mnih et al., “Human-level control through deep reinforcement learning” (DQN), Nature 2015
- Schulman et al., “Proximal Policy Optimization Algorithms”, arXiv 2017
- Christiano et al., “Deep RL from human preferences”, NeurIPS 2017; Ouyang et al., InstructGPT, NeurIPS 2022
- Rafailov et al., “Direct Preference Optimization”, NeurIPS 2023
- Shao et al., DeepSeekMath (GRPO), 2024; DeepSeek-AI, DeepSeek-R1, 2025
- N. Lambert et al., RLHF tutorial, ICML 2023; Tülu 3, Ai2 2024
- Slides adapted from E. Xing (CMU), G. Hulten, P. Bodik; Stanford CS224N W26 L08; CMU 11-711 S25 L23; M. Grootendorst, A Visual Guide to Reasoning LLMs