# Policy gradient

Simplicity at the cost of variance

**Cathy Wu**

6.7920: Reinforcement Learning: Foundations and Methods

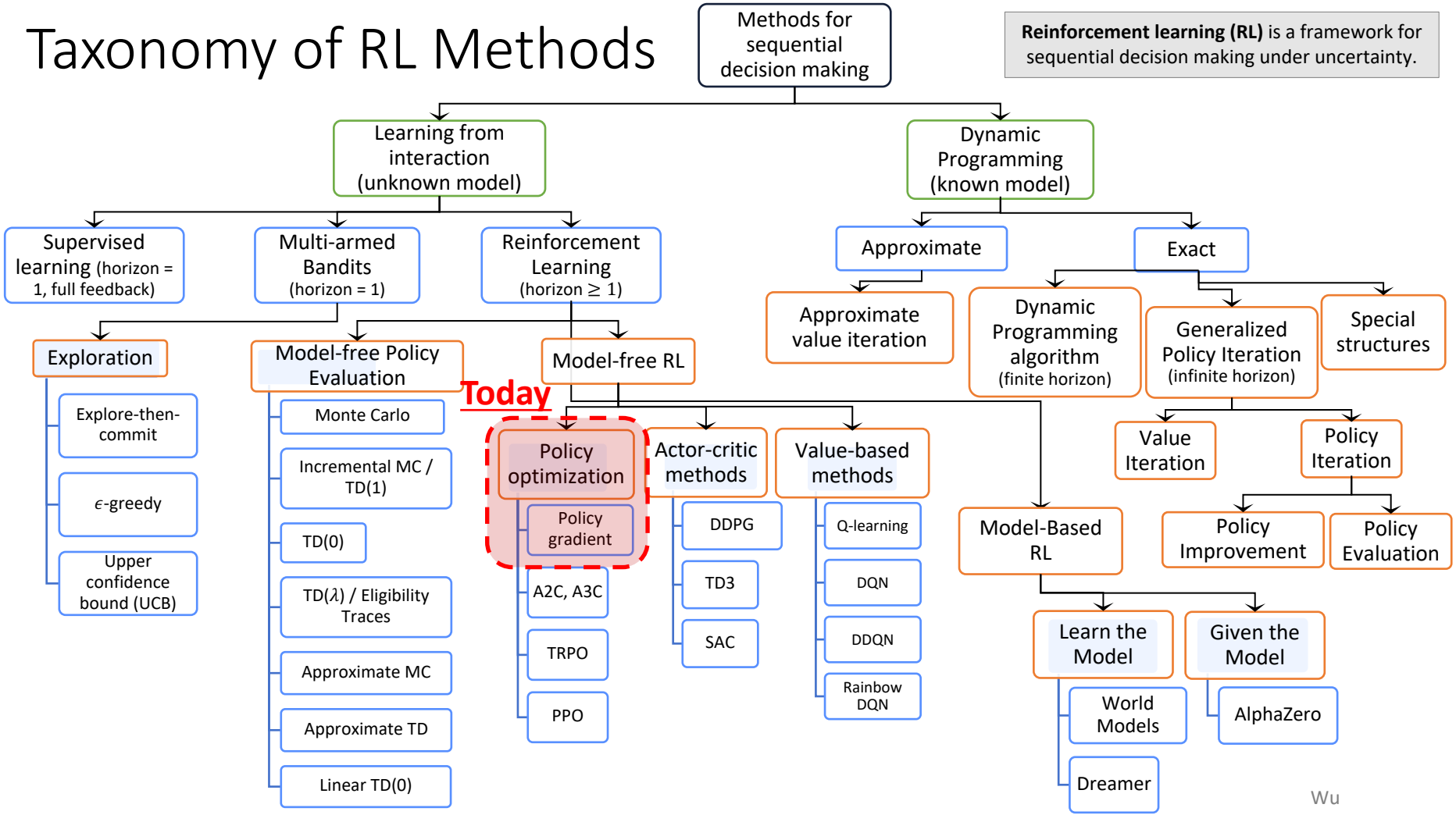
# Readings

1. *Josh Achiam. Spinning Up. Part 3: Intro to Policy Optimization. OpenAI, 2018.*
2. *NDP §6.1: Generic issues – from parameters to policies*
3. *SB Chapter 13: Policy Gradient Methods*

# Outline

1. From Policy Iteration to Policy Search
2. Representing a policy (discrete and continuous!)
3. Policy gradient methods
4. REINFORCE
5. Variance reduction (temporal structure and baselines)

# Taxonomy of RL Methods




**Reinforcement learning (RL)** is a framework for sequential decision making under uncertainty.

Wu

# *Function approximation*

**Last time:** Add function approximation to **value iteration**

Now let's try to add function approximation to **policy iteration**.

# Outline

1. **From Policy Iteration to Policy Search**
2. Representing a policy (discrete and continuous!)
3. Policy gradient methods
4. REINFORCE
5. Variance reduction (temporal structure and baselines)
6. RL for Large Language Models

# Policy Iteration: Recap

Let $\pi_0$ be an arbitrary stationary policy.

**while** $k = 1, \dots, K$ **do**

Policy Evaluation: given $\pi_k$ compute $V_k = V^{\pi_k}$

Policy Improvement: find $\pi_{k+1}$ that is better than $\pi_k$

- e.g. compute the *greedy* policy:

$$\pi_{k+1}(s) \in \arg\max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_y p(y|s,a) V^{\pi_k}(y) \right\}$$

**return** the last policy $\pi_K$

**end**

---

- Convergence is finite and monotonic [Bertsekas, 2007] (in exact settings)

? Issues: Function approximation for $V^{\pi_k} \Longrightarrow$ Does it still converge?

Continuous Actions?

# Approximate Policy Iteration with $Q$ Functions

Recall the state-action cost-to-go function: $Q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) Q_\pi\big(s', \pi(s')\big)$

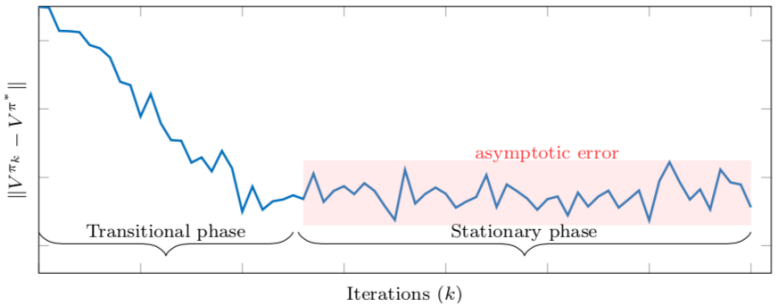
**Approximate PI:**

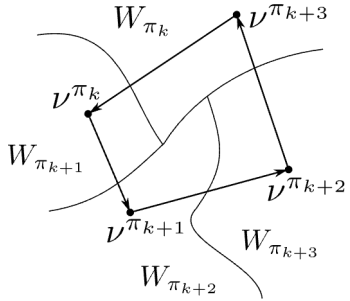
- For $k = 0, 1, 2, \ldots$
  1. Approximate the value under $\pi_k$: $Q_{\theta_k} \approx Q_{\pi_k}$
  2. Solve for an improved policy

$$\pi_{k+1}(s) \in \underset{a \in A(s)}{\operatorname{argmin}} Q_{\theta_k}(s,a) \qquad \forall s \in \mathcal{S}$$

$Q_{\pi_k}$ can be approximated by either TD or Monte Carlo methods.

Same story as fitted Q-iteration. No longer guaranteed to converge.



# Policy Gradient as Policy Update

- Approximate a stochastic policy directly using function approximation

$$\pi_\theta : S \to \mathcal{P}(\mathcal{A}) \text{ with } \theta \in \mathbb{R}^d$$

- Let $V(\pi_\theta)$ denote the policy performance of policy $\pi_\theta$
- Policy optimization problem

$$\max_{\pi_\theta} V(\pi_\theta)$$

Approximate Policy Iteration

$$\pi_{\theta_{k+1}} = \arg\max_{\pi_\theta} Q^{\pi_\theta}\left(s, \pi_\theta(s)\right)$$

Unstable (fast)

No convergence

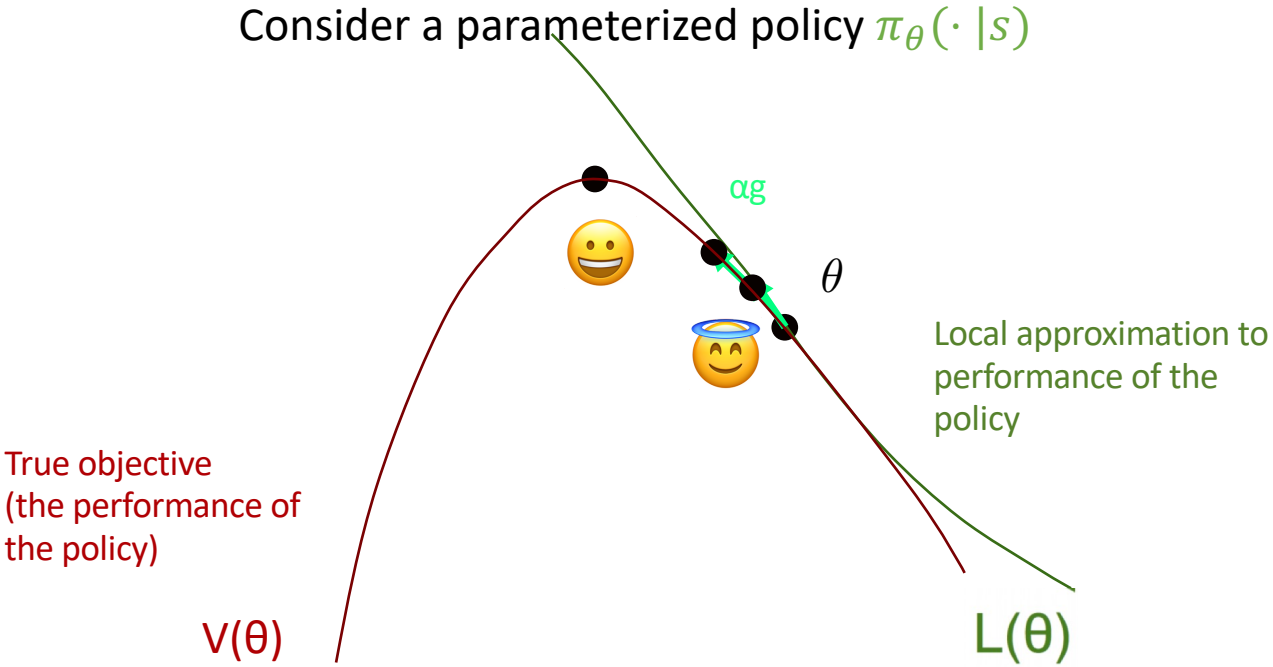
Policy Gradient

$$\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta V(\theta_k)$$

Smooth, fine control (slow)

Convergence to local optima

# Policy gradient = gradient ascent for MDPs




Adapted from Matteo Pirotta

# Policy Gradient = gradient ascent for MDPs

$$V(\pi_{\theta_k}) = \mathbb{E}\left[\sum_{t=0}^{T-1} r_t | \pi_{\theta_k}, M\right] = \mathbb{E}_{\tau \sim \mathbb{P}(\tau | \pi_{\theta_k}, M)}[\mathcal{R}(\tau)]$$

Policy Gradient

$$\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta V(\theta_k)$$

1. **How do we compute $\nabla_\theta V(\theta)$?**
2. How quickly do we update (i.e. $\alpha_k$)?

REINFORCE, variance reduction, baselines, generalized advantage estimation (GAE)

NPG, TRPO, PPO

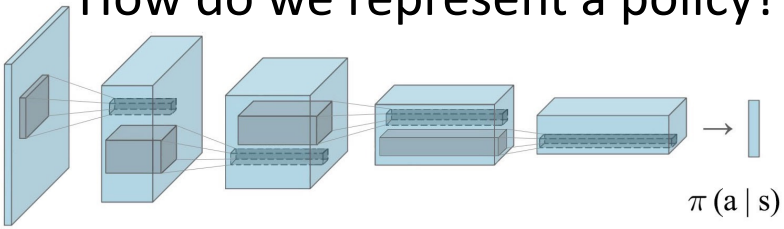
# Outline

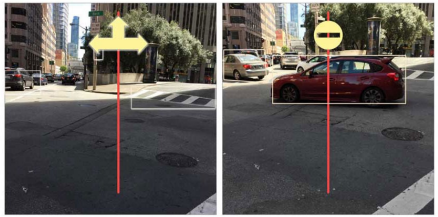
1. From Policy Iteration to Policy Search
2. **Representing a policy (discrete and continuous!)**
3. Policy gradient methods
4. REINFORCE
5. Variance reduction (temporal structure and baselines)
6. RL for Large Language Models

# Example: Parameterized Policy



How do we represent a policy?



Normal Policy

$$\pi(a|s) = \frac{1}{\sigma_\omega(s)\sqrt{2\pi}} e^{-\frac{(a-\mu_\theta(s))^2}{2\sigma_\omega^2(s)}}$$

Continuous actions

Gibbs (softmax) Policy

$$\pi(a|s) = \frac{e^{\mathcal{K}Q_\theta(s,a)}}{\sum_{a' \in \mathcal{A}} e^{\mathcal{K}Q_\theta(s,a')}}$$

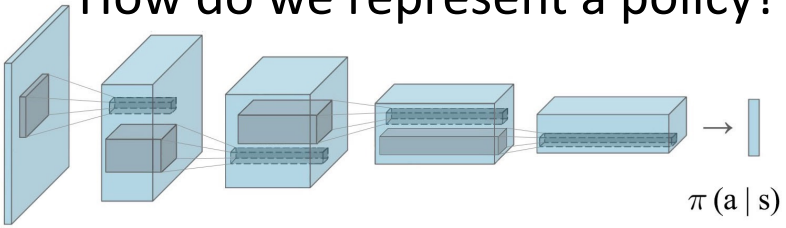
Discrete actions

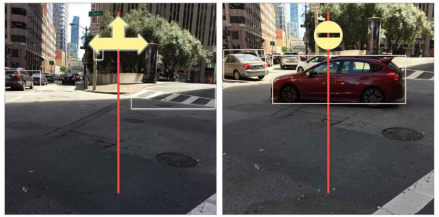
Differentiable! → autodiff via PyTorch

# Example: Parameterized Policy



How do we represent a policy?



Normal Policy

$$\pi(a|s) = \frac{1}{\sigma_\omega(s)\sqrt{2\pi}} e^{-\frac{(a-\mu_\theta(s))^2}{2\sigma_\omega^2(s)}}$$

Then:

$$\nabla_\theta \log \pi(a|s) = \frac{(a-\mu_\theta(s))}{\sigma_\omega^2(s)} \nabla_\theta \mu_\theta(s)$$

$$\nabla_\omega \log \pi(a|s) = \frac{(a-\mu_\theta(s))^2 - \sigma_\omega^2(s)}{\sigma_\omega^3(s)} \nabla_\omega \mu_\omega(s)$$

Continuous actions

Gibbs (softmax) Policy

$$\pi(a|s) = \frac{e^{\mathcal{K} Q_\theta(s,a)}}{\sum_{a' \in \mathcal{A}} e^{\mathcal{K} Q_\theta(s,a')}}$$

Then:

$$\nabla_\theta \log \pi(a|s) = \mathcal{K} \nabla_\theta Q_\theta(s,a) - \mathcal{K} \sum_{a' \in \mathcal{A}} \pi(a'|s) \nabla_\theta Q_\theta(s,a')$$

Discrete actions

# Outline

1. From Policy Iteration to Policy Search
2. Representing a policy (discrete and continuous!)
3. **Policy gradient methods**
4. REINFORCE
5. Variance reduction (temporal structure and baselines)
6. RL for Large Language Models

*Assume: finite-horizon setting*

Discount $\gamma$ excluded to simplify notation.

# Policy Gradient (Finite-Horizon)

Given an MDP $M = (\mathcal{S}, \mathcal{A}, p, r, T, \mu)$ and a policy $\pi_{\theta_0}$. For k = 1,2,…

1. Use $\pi_{\theta_k}$ to collect data $\tau$.
2. Use $\tau$ to approximate gradient of:

Maximizing this is ultimately what we desire

$$V(\pi_{\theta_k}) = \mathbb{E}\left[\sum_{t=0}^{T-1} r_t | \pi_{\theta_k}, M\right] = \mathbb{E}_{\tau \sim \mathbb{P}(\tau | \pi_{\theta_k}, M)}[\mathcal{R}(\tau)]$$

where

- $\mu$ is an initial state distribution
- $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$ (includes terminal reward) is a trajectory
- $\mathcal{R}(\tau)$ its return (sum of rewards).

Main issue: MDP is a complex object to differentiate through, i.e. $\nabla_\theta \mathbb{P}(\tau | \pi_\theta, M)$. (Example: Frozen Lake)

3. Update $\theta_{k+1} = \theta_k + \alpha_k \widehat{\nabla_\theta V}(\pi_{\theta_k})$

How?

# Example: Frozen Lake (Gymnasium)

- Aim:
  - Make it to the goal
  - Don’t fall into the holes
- Slippery (stochastic actions)
- Observation: current location



# Policy Gradient (Finite-Horizon)

**Policy Gradient Theorem** [Williams, 1992; Sutton et al., 2000]

For any finite-horizon MDP $M = (\mathcal{S}, \mathcal{A}, p, r, T, \mu)$ and differentiable policy $\pi_\theta$

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}_{\tau \sim \mathbb{P}(\cdot|\pi, M)} \left[ R(\tau) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s_t, a_t) \right]$$

Gradient is now on the inside! We can differentiate through (differentiable) policies.

- Model-free! Why?
- Compare: taking gradient through trajectory-space is difficult

$$\nabla_\theta V(\pi_\theta) = \nabla_\theta \mathbb{E}_\tau[R(\tau)] = \nabla_\theta \int \mathbb{P}(\tau|\pi_\theta, M) R(\tau) d\tau$$

# Proof

- The objective is an expectation. Want to compute the gradient w.r.t. $\theta$ (simplify notation from: $V(\pi_\theta)$ to $V(\theta)$). First, bring the gradient to the inside.

$$\nabla_\theta V(\theta) = \nabla_\theta \mathbb{E}_\tau[R(\tau)] = \nabla_\theta \int \mathbb{P}(\tau|\pi_\theta, M) R(\tau) d\tau$$

Log trick

$$\nabla_\theta \log \mathbb{P}(\tau|\pi_\theta, M) = \frac{\nabla_\theta \mathbb{P}(\tau|\pi_\theta, M)}{\mathbb{P}(\tau|\pi_\theta, M)}$$

$$= \int \nabla_\theta \mathbb{P}(\tau|\pi_\theta, M) R(\tau) d\tau$$

$$= \int \mathbb{P}(\tau|\pi_\theta, M) \nabla_\theta \log \mathbb{P}(\tau|\pi_\theta, M) \, R(\tau) d\tau$$

$$= \mathbb{E}_\tau[R(\tau) \nabla_\theta \log \mathbb{P}(\tau|\pi_\theta, M)]$$

- Last expression is an unbiased gradient estimator
  Just sample $\tau_t \sim \mathbb{P}(\tau|\pi_\theta, M)$, and compute $\hat{g}_t = R(\tau_t) \nabla_\theta \log \mathbb{P}(\tau_t|\pi_\theta, M)$
- Issue: Need to be able to compute & differentiate the density $\mathbb{P}(\tau|\pi_\theta, M)$ w.r.t $\theta$

# Proof

Likelihood (with stochastic policies)

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

$$\log\mathbb{P}(\tau|\pi_\theta, M) = \log\mu(s_0) + \sum_{t=0}^{T-1}\log\pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t)$$

$$\nabla_\theta\log\mathbb{P}(\tau|\pi_\theta, M) = \nabla_\theta\log\mu(s_0) + \sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_t|s_t) + \nabla_\theta\log p(s_{t+1}|s_t, a_t)$$

0 0

→ model free

# Alternative proof: likelihood rescaling

- Interested in policy gradient: $\nabla_\Delta V(\theta+\Delta)|_{\Delta=0}$
- Likelihood rescaling

$$V(\theta+\Delta) = \mathbb{E}_{\tau(\theta)}\left[R(\tau(\theta))\frac{\prod_t \pi_{\theta+\Delta}(a_t|s_t)}{\prod_t \pi_\theta(a_t|s_t)}\right]$$

- Apply chain rule to get

$$\nabla_\Delta V(\theta+\Delta)\Big|_{\Delta=0} = \mathbb{E}_{\tau(\theta)}\left[R(\tau(\theta))\sum_t \frac{\nabla \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)}\right]$$
$$= \mathbb{E}_\tau[R(\tau)\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

# Policy Gradient (Finite-Horizon)

**Policy Gradient Theorem** [Williams, 1992; Sutton et al., 2000]

For any finite-horizon MDP $M = (\mathcal{S}, \mathcal{A}, p, r, T, \mu)$ and differentiable policy $\pi_\theta$

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}_{\tau \sim \mathbb{P}(\cdot|\pi, M)} \left[ R(\tau) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s_t, a_t) \right]$$

Gradient is now on the inside! We can differentiate through (differentiable) policies.

- Model-free! Why?
- Compare: taking gradient through trajectory-space is difficult

$$\nabla_\theta V(\pi_\theta) = \nabla_\theta \mathbb{E}_\tau[R(\tau)] = \nabla_\theta \int \mathbb{P}(\tau|\pi_\theta, M) R(\tau) d\tau$$

# Outline

1. From Policy Iteration to Policy Search
2. Representing a policy (discrete and continuous!)
3. Policy gradient methods
4. **REINFORCE**
5. Variance reduction (temporal structure and baselines)
6. RL for Large Language Models

# REINFORCE [Williams, 1992]

1. Let $\pi_{\theta_1}$ be an arbitrary policy.
2. At each iteration $k = 1, \dots, K$
   - Sample $m$ trajectories $\tau_i = (s_0, a_0, r_0, s_1, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$ following $\pi_k$
   - Compute unbiased gradient estimate:

$$\widehat{\nabla_\theta V}(\pi_{\theta_k}) = \frac{1}{m}\sum_{i=1}^{m}\left(\sum_{t=0}^{T-1} r_t^i\right)\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_{\theta_k}(a_t^i | s_t^i)\right)$$

   - Update parameters:

$$\theta_{k+1} = \theta_k + \alpha_k \widehat{\nabla_\theta V}(\pi_{\theta_k})$$

Monte Carlo approximation of policy gradient

3. Return last policy $\pi_{\theta_K}$

# REINFORCE [Williams, 1992]

1. Let $\pi_{\theta_1}$ be an arbitrary policy.
2. At each iteration $k = 1, \dots, K$
   - Sample $m$ trajectories $\tau_i = (s_0, a_0, r_0, s_1, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$ following $\pi_k$
   - Compute unbiased gradient estimate:

$$\widehat{\nabla_\theta V}(\pi_{\theta_k}) = \frac{1}{m}\sum_{i=1}^{m}\left(\sum_{t=0}^{T-1} r_t^i\right)\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_{\theta_k}(a_t^i | s_t^i)\right)$$

   - Update parameters:

$$\theta_{k+1} = \theta_k + \alpha_k \widehat{\nabla_\theta V}(\pi_{\theta_k})$$

Monte Carlo approximation of policy gradient

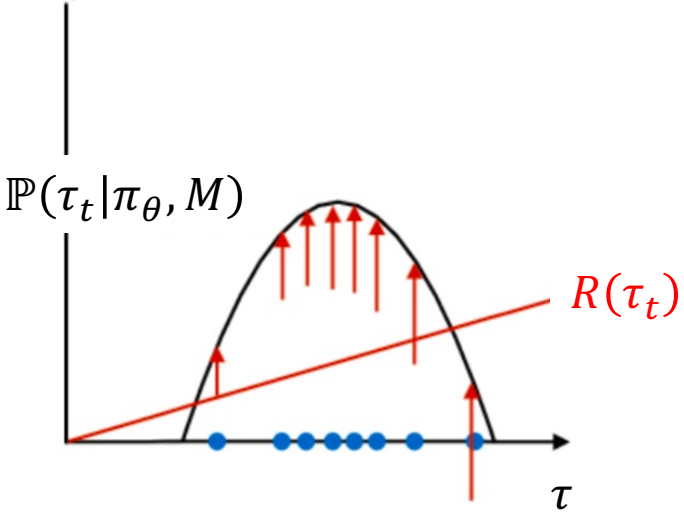
3. Return last policy $\pi_{\theta_K}$



# REINFORCE as Supervised Learning

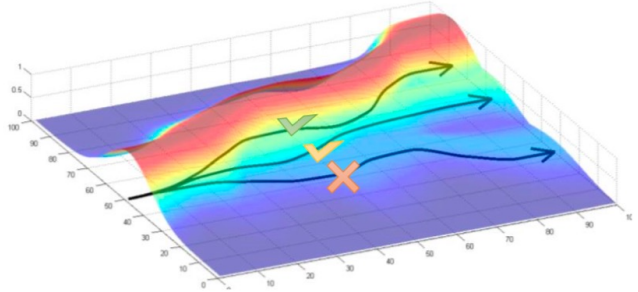
$$\hat{g}_t = R(\tau_t)\nabla_\theta \log \mathbb{P}(\tau_t|\pi_\theta, M)$$

- $R(\tau_t)$ measures how good is sample $\tau_t$
- Moving in the direction of $\hat{g}_t$ pushes up the log probability of the sample in proportion to how good it is.




Interpretation: uses good trajectories as supervised examples

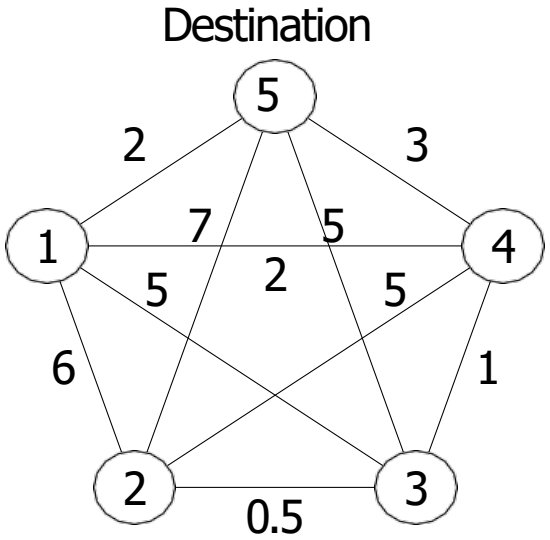
- Like maximum likelihood in supervised learning
- Good stuff are made more likely while bad less
- Trial and Error approach



From "CS 294-112: Deep Reinforcement Learning" slides by S. Levine

# *Dynamic programming vs policy gradient*

How would policy gradient solve shortest path?




Destination is node 5.

# REINFORCE

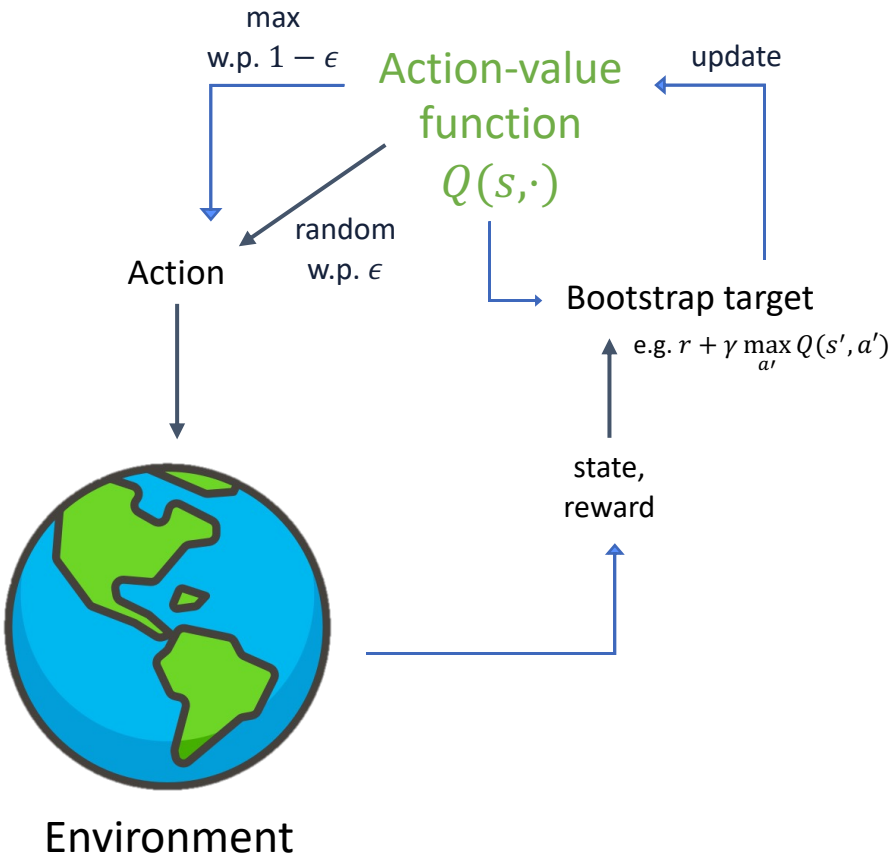
**Pros**

- Easy to compute
- Does not use Markov property!
- Can be used in partially observable MDPs without modification

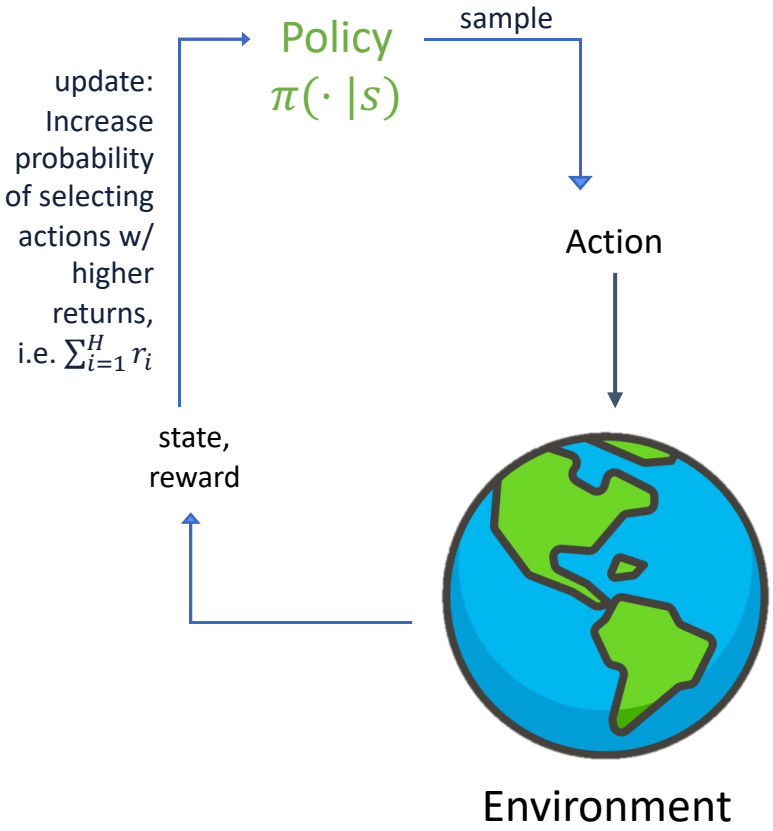
**Issues**

- Use a MC estimate of $Q(s, a)$
- It has possibly a very large variance
- Needs many samples to converge

# Value-based methods



# Policy-based methods

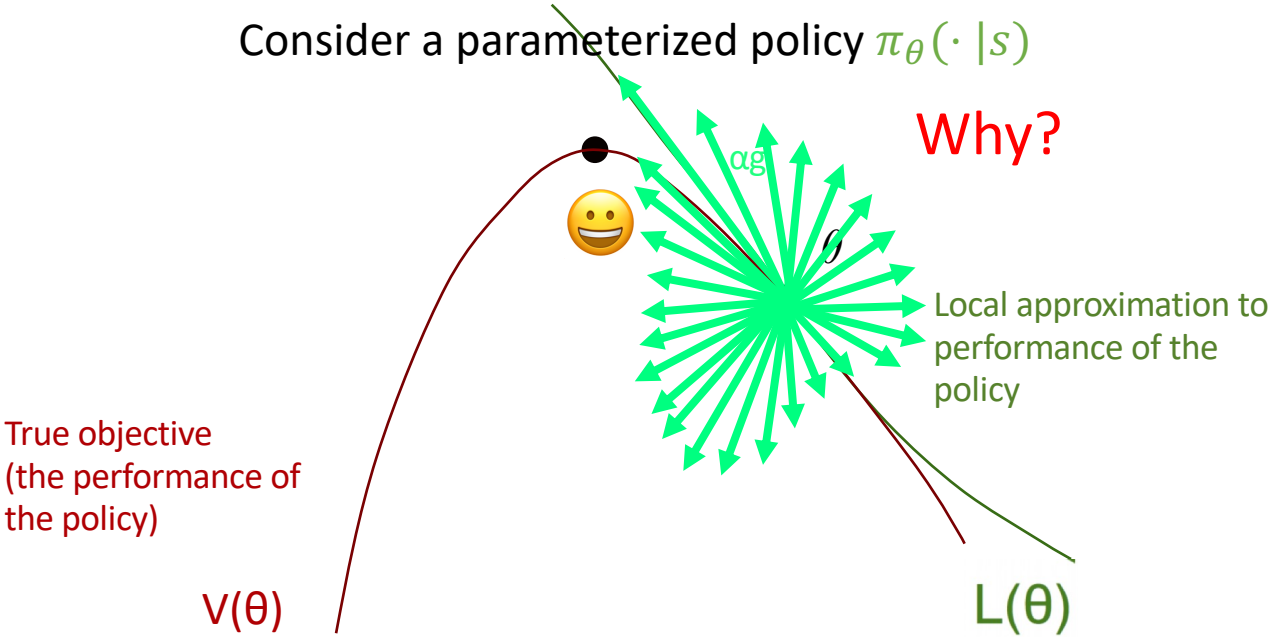


# Outline

1. From Policy Iteration to Policy Search
2. Representing a policy (discrete and continuous!)
3. Policy gradient methods
4. REINFORCE
5. **Variance reduction (temporal structure and baselines)**
6. RL for Large Language Models

# Key challenge: Policy gradient has high variance

Consider a parameterized policy $\pi_\theta(\cdot|s)$




$$\widehat{\nabla_\theta V}(\pi_{\theta_k}) = \frac{1}{m}\sum_{i=1}^{m}\left(\sum_{t=0}^{T-1} r_t^i\right)\left(\sum_{t=0}^{T-1}\nabla_\theta \log \pi_{\theta_k}(a_t^i|s_t^i)\right)$$

Adapted from Matteo Pirotta

# Policy Gradient: Temporal Structure (Causality)

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T-1} r_{t'}\right]$$

**Discuss**: Why does this help with variance?

Because $\forall t$ :

Actions don't affect past rewards

$$\mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s_t) \sum_{t'=0}^{t-1} r_i \,|\tau_{0:t-1}\right] = \left(\sum_{t'=0}^{t-1} r_i\right) \int \pi_\theta(s_t, a) \nabla_\theta \log \pi_\theta(a|s_t) da$$

$$= \left(\sum_{t'=0}^{t-1} r_i\right) \int \nabla_\theta \pi_\theta(a|s_t) da$$

$$= \left(\sum_{t'=0}^{t-1} r_i\right) \nabla_\theta \underbrace{\int \pi_\theta(a|s_t) da}_{:=1} = 0$$

In literature known as G(PO)MDP [Peters and Schaal, 2008b].

# Policy Gradient: Baseline

- Further reduce the variance by introducing a baseline $b(s)$

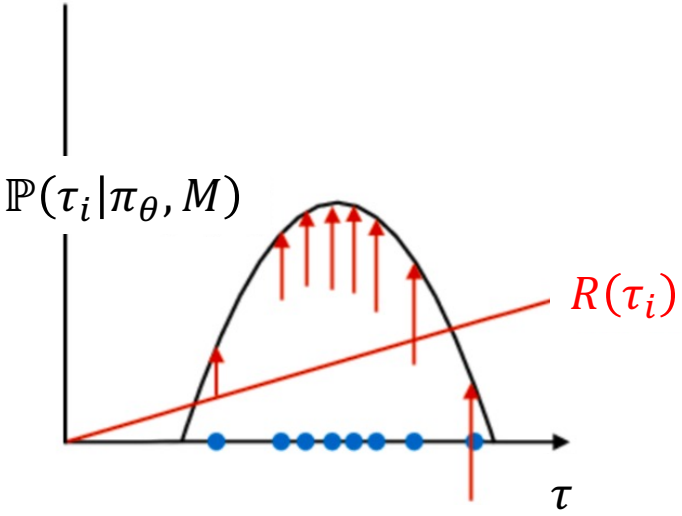
$$\nabla_\theta V(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)\right]$$

- The gradient estimate is still unbiased.
- Proof: State-dependent baselines do not introduce bias (zero mean).

# Solution: Baseline

Regular policy gradient

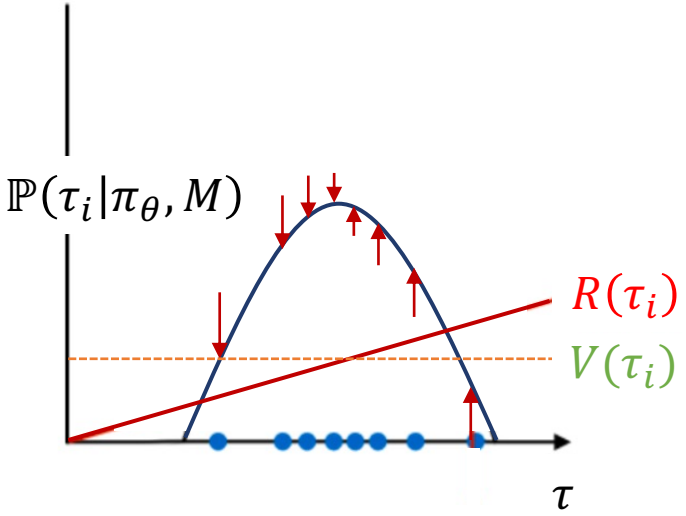
- $\hat{g}_i = R(\tau_i)\nabla_\theta \log \mathbb{P}(\tau_i|\pi_\theta, M)$




Encourage all trajectories

Policy gradient with baseline

- $\hat{g}_i = (R(\tau_i) - V(\tau_i))\nabla_\theta \log \mathbb{P}(\tau_i|\pi_\theta, M)$




Encourage trajectories that are better than average

# Variance reduction via baseline?

- Baseline

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)\right]$$

- "Near optimal choice" that minimize the variance is the expected sum of returns:

$$b^\star(s) \approx \mathbb{E}\left[\sum_{t=0}^{T-1} r_t | s_0 = s, \pi_\theta, M\right] = V^{\pi_\theta}(s)$$

  - *Interpretation*: increase the log probability of an action $a_t$ proportionally to how much returns are **better than expected** (relative values).

- Intuition: To reduce variance, try to maximize the covariance between x and y

$$\mathrm{Var}(x - y) = \mathrm{Var}(x) - 2\mathrm{Cov}(x, y) + \mathrm{Var}(y)$$

# Optimal Baseline Derivation

$$\nabla_{\theta_i} V(\pi_\theta) = \mathbb{E}_\tau\Big[\underbrace{\nabla_{\theta_i} \log \mathbb{P}(\tau|\pi_\theta)}_{:= g(\tau)}(R(\tau) - b)\Big]$$

$$\text{Var} = \mathbb{E}_\tau[(g(\tau)(R(\tau) - b))^2] - (\mathbb{E}_\tau[g(\tau)(R(\tau) - b)])^2$$
$$\Longrightarrow \mathbb{E}_\tau[g(\tau)R(\tau)]^2$$

[Baseline is unbiased in expectation]

$$\frac{\partial}{\partial \text{b}} \text{Var} = \frac{\partial}{\partial \text{b}} \mathbb{E}_\tau[g(\tau)^2(R(\tau) - b)^2]$$
$$= \frac{\partial}{\partial \text{b}} \mathbb{E}_\tau[g(\tau)^2 R(\tau)^2] - 2\frac{\partial}{\partial \text{b}} \mathbb{E}_\tau[g(\tau)^2 R(\tau) b] + \frac{\partial}{\partial \text{b}} \mathbb{E}_\tau[b^2 g(\tau)^2]$$

($\frac{\partial}{\partial \text{b}} \mathbb{E}_\tau[g(\tau)^2 R(\tau)^2]$ → 0)

$$\Longrightarrow b^\star(\tau) = \frac{\mathbb{E}_\tau[g(\tau)^2 R(\tau)]}{\mathbb{E}_\tau[g(\tau)^2]}$$

Expected return weighted by the magnitude of the gradient.

# Aside / Reminder: Sampling settings

- Learning with generative model. A black-box simulator $f$ of the environment is available. Given $(s, a)$,
$$f(s,a) = \{s', r'\} \text{ with } s' \sim p(\cdot\,|s,a), r' = r(s,a)$$
- Episodic learning. Multiple trajectories can be repeatedly generated from some initial states and terminating when a reset condition is achieved:
$$\left(s_{0,i}, s_{1,i}, \ldots, s_{T_i,i}\right)_{i=1}^{n}$$
- Online learning. At each time $t$ the agent is at state $s_t$, it takes action $a_t$, it observes a transition to state $s_{t+1}$, and it receives a reward $r_{t+1}$. We assume that $s_{t+1} \sim p(\cdot\,|s_t, a_t)$ and $r_{t+1} = r(s_t, a_t)$ (i.e., MDP assumption). No reset.

# REINFORCE Leave-One-Out (RLOO)

- Setting: Learning with generative model
- Insight: Can compute an (unbiased) Monte-Carlo estimate of $V^{\pi_\theta}(s)$
- Strategy: Reuse the sampled returns for data and baseline estimates
- Generate $K$ trajectories from each state $s$

$$b_{-k}(s) = \frac{1}{K-1} \sum_{i=1, i \neq k}^{K} \sum_{t=t'}^{T-1} r_{t,i}$$

- The gradient estimator then becomes:

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}\left[ \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_{t,k} | s_{t,k}) \left( \sum_{t'=t}^{T-1} r_{t',k} - b_{-k}(s_t) \right) \right]$$

- Issue: Exponential branching
- Typical usage (structured prediction, e.g., vehicle routing problems, large language models): Applied only on initial states $s_0$.

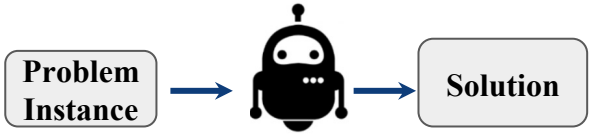
W. Kool, H. van Hoof, and M. Welling, "Buy 4 reinforce samples, get a baseline for free!" ICLR Workshop on Deep RL Meets Structured Prediction, 2019.
A. Ahmadian *et al.*, "Back to basics: Revisiting REINFORCE style optimization for learning from human feedback in LLMs," in *ACL*, 2024.

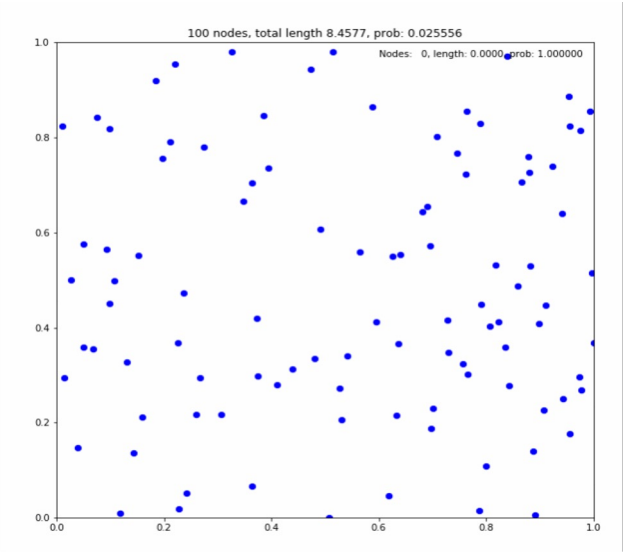
# Structured prediction

- Machine learning aims to learn a mapping from input space $\mathcal{X}$ to some output space $\mathcal{Y}$.
- Examples
  - Binary classification: $\mathcal{Y} = \{0,1\}$
  - Multi-class classification: $\mathcal{Y} = \{0, \dots, k\}$
- Structured prediction is when the output space is large and cannot be decomposed due to underlying dependencies ("structure")
- Examples
  - Image semantic segmentation: $|\mathcal{Y}| = k^{hw}$, where the image is $h \times w$ pixels
  - Neural combinatorial optimization
  - Language modeling

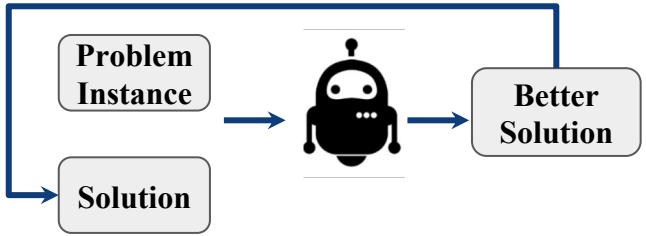
# RL for solving combinatorial optimization

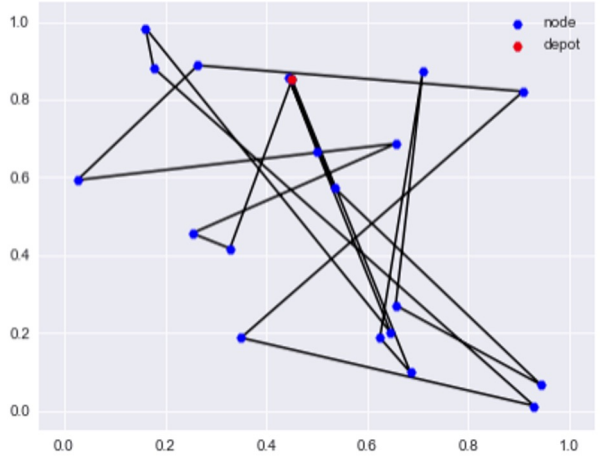
## Construction methods




## Improvement methods



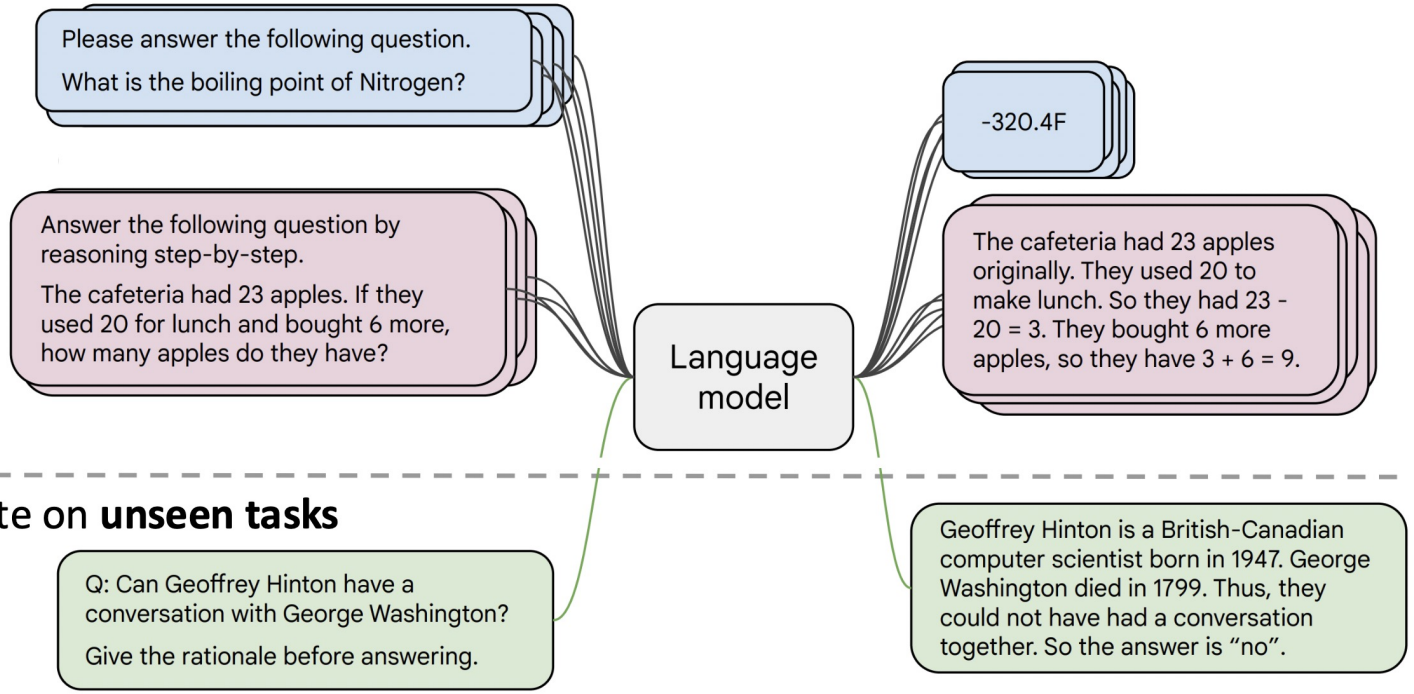

Slide courtesy Yining Ma

# Outline

1. From Policy Iteration to Policy Search
2. Representing a policy (discrete and continuous!)
3. Policy gradient methods
4. REINFORCE
5. Variance reduction (temporal structure and baselines)
6. **RL for Large Language Models**

# Overall problem: Instruction fine-tuning

- **Collect examples** of (instruction, output) pairs across many tasks and finetune an LM

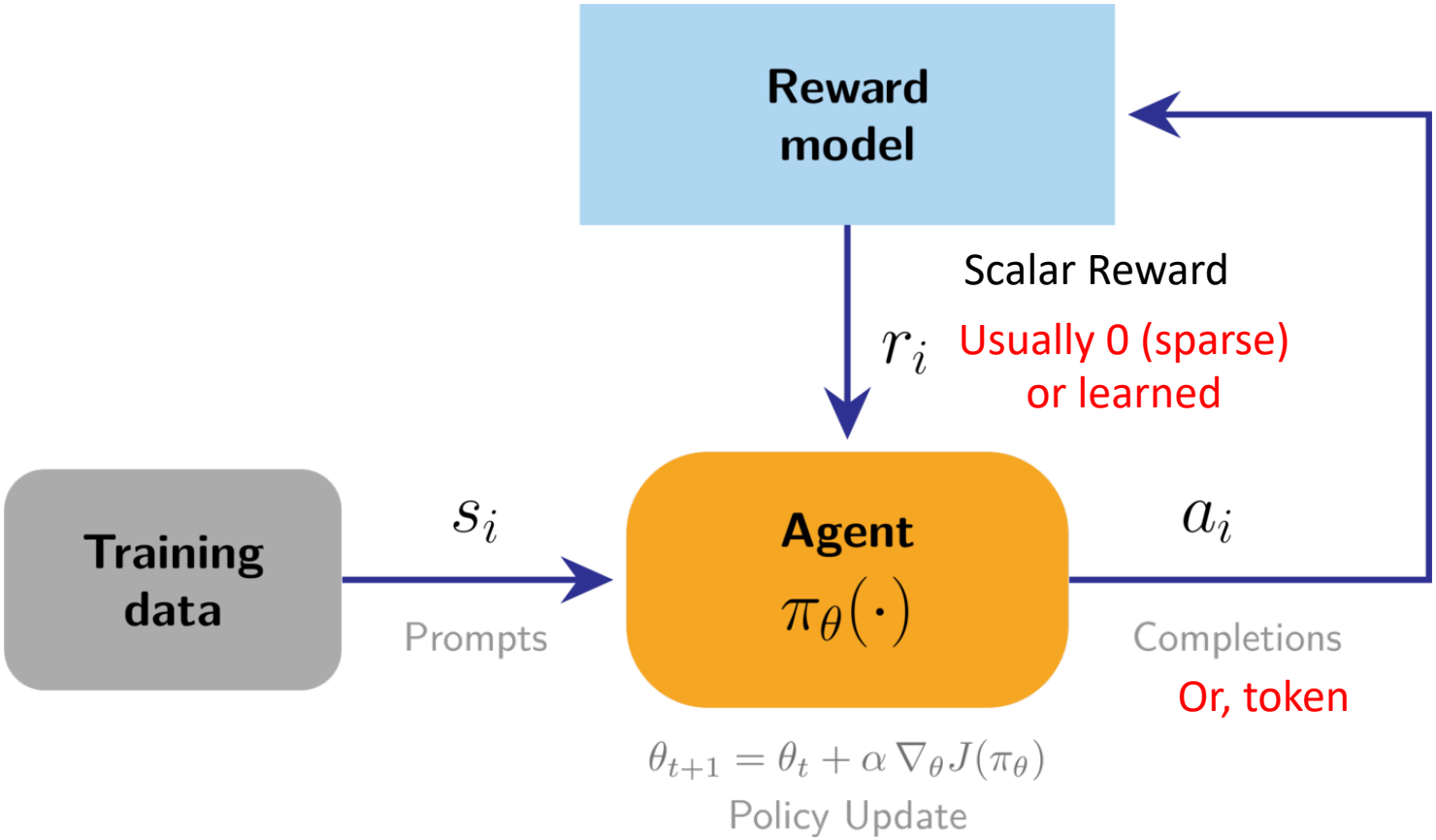



- Evaluate on **unseen tasks**

[FLAN-T5; Chung et al., 2022]

Slide courtesy Archit Sharma

# Standard RLHF loop (human feedback)




Lambert, Nathan. "Reinforcement Learning from Human Feedback." 2025, rlhfbook.com/c/03-training-overview.

# Problem Setup: Optimizing for human preferences

- Let's say we were training a language model on some task (e.g. summarization).
- For an instruction $x$ and a LM sample $y$, imagine we had a way to obtain a *human reward* of that summary: $R(x, y) \in \mathbb{R}$, higher is better.

$x$

```
SAN FRANCISCO,
California (CNN) --
A magnitude 4.2
earthquake shook the
San Francisco
...
overturn unstable
objects.
```

$y_1$

```
An earthquake hit
San Francisco.
There was minor
property damage,
but no injuries.
```

$R(x, y_1) = 8.0$

$y_2$

```
The Bay Area has
good weather but is
prone to
earthquakes and
wildfires.
```

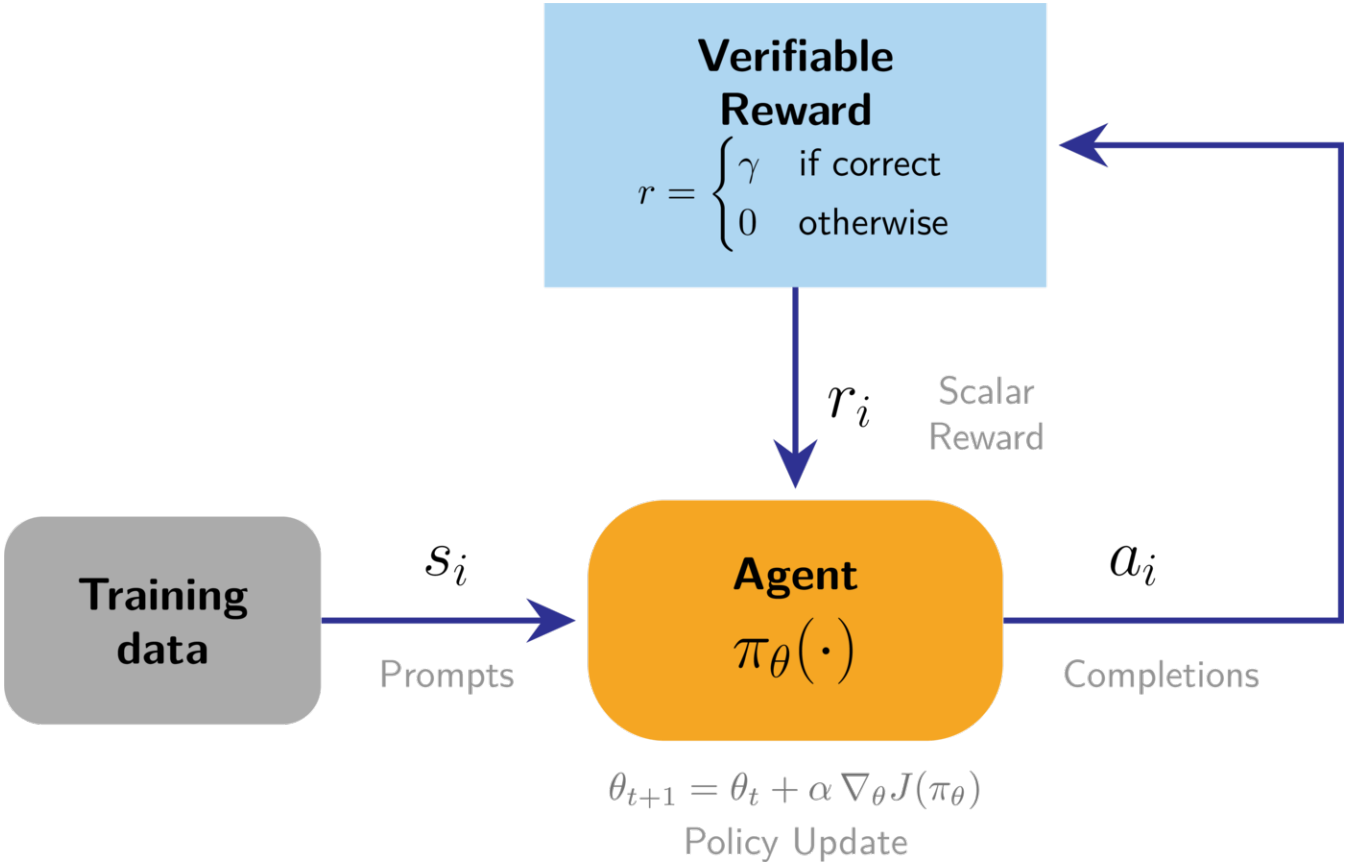
$R(x, y_2) = 1.2$

- Now we want to maximize the expected reward of samples from our LM:

$$\mathbb{E}_{\hat{y} \sim p_\theta(y \mid x)}[R(x, \hat{y})]$$

Slide courtesy Archit Sharma

# Standard RLVR loop (verifiable reward)




Lambert, Nathan. "Reinforcement Learning from Human Feedback." 2025, rlhfbook.com/c/07-reasoning.

# Problem Setup: LLM Math Reasoning Problems

Initial state

**Problem:** Suppose $a$ and $b$ are positive real numbers with $a > b$ and $ab = 8$. Find the minimum value of $\frac{a^2+b^2}{a-b}$.

**Ground truth solution:** We can write $\frac{a^2+b^2}{a-b} = \frac{a^2+b^2-2ab+16}{a-b} = \frac{(a-b)^2+16}{a-b} = a - b + \frac{16}{a-b}$. By AM-GM, $a - b + \frac{16}{a-b} \geq 2\sqrt{(a-b) \cdot \frac{16}{a-b}} = 8$. Equality occurs when $a - b = 4$ and $ab = 8$. We can solve these equations to find $a = 2\sqrt{3} + 2$ and $b = 2\sqrt{3} - 2$. Thus, the minimum value is $\boxed{8}$.

Steps = actions

reward = 1 if answer is correct

**Intuition:** Sparse-reward MDP with deterministic dynamics

Slide courtesy Aviral Kumar

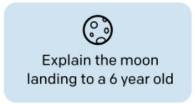
# InstructGPT: scaling up RLHF to 30k tasks

**30k tasks!**

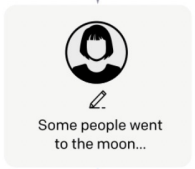
Step 1

**Collect demonstration data, and train a supervised policy.**

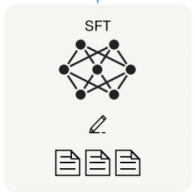
A prompt is sampled from our prompt dataset.




A labeler demonstrates the desired output behavior.




This data is used to fine-tune GPT-3 with supervised learning.




Step 2

**Collect comparison data, and train a reward model.**

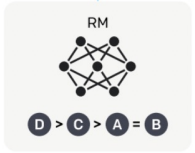
A prompt and several model outputs are sampled.




A labeler ranks the outputs from best to worst.




This data is used to train our reward model.




Step 3

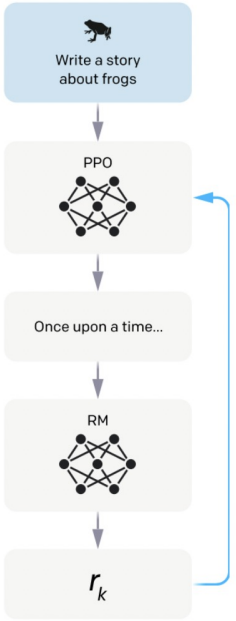
**Optimize a policy against the reward model using reinforcement learning.**

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.




[Ouyang et al., 2022]

# InstructGPT: scaling up RLHF to 30k tasks

**Tasks collected from labelers:**

- **Plain:** We simply ask the labelers to come up with an arbitrary task, while ensuring the tasks had sufficient diversity.
- **Few-shot:** We ask the labelers to come up with an instruction, and multiple query/response pairs for that instruction.
- **User-based:** We had a number of use-cases stated in waitlist applications to the OpenAI API. We asked labelers to come up with prompts corresponding to these use cases.

| Use-case | Prompt |
|---|---|
| Brainstorming | List five ideas for how to regain enthusiasm for my career |
| Generation | Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home. |

[Ouyang et al., 2022]

# InstructGPT

**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.
```

# InstructGPT

**PROMPT** *Write a short poem about a wise frog.*

**COMPLETION**

GPT-3

```
Write a short story in which a character has two different names.

Write a short story in which you try to get something back that you
have lost.

Write a short story in which a character has a bad dream.
```

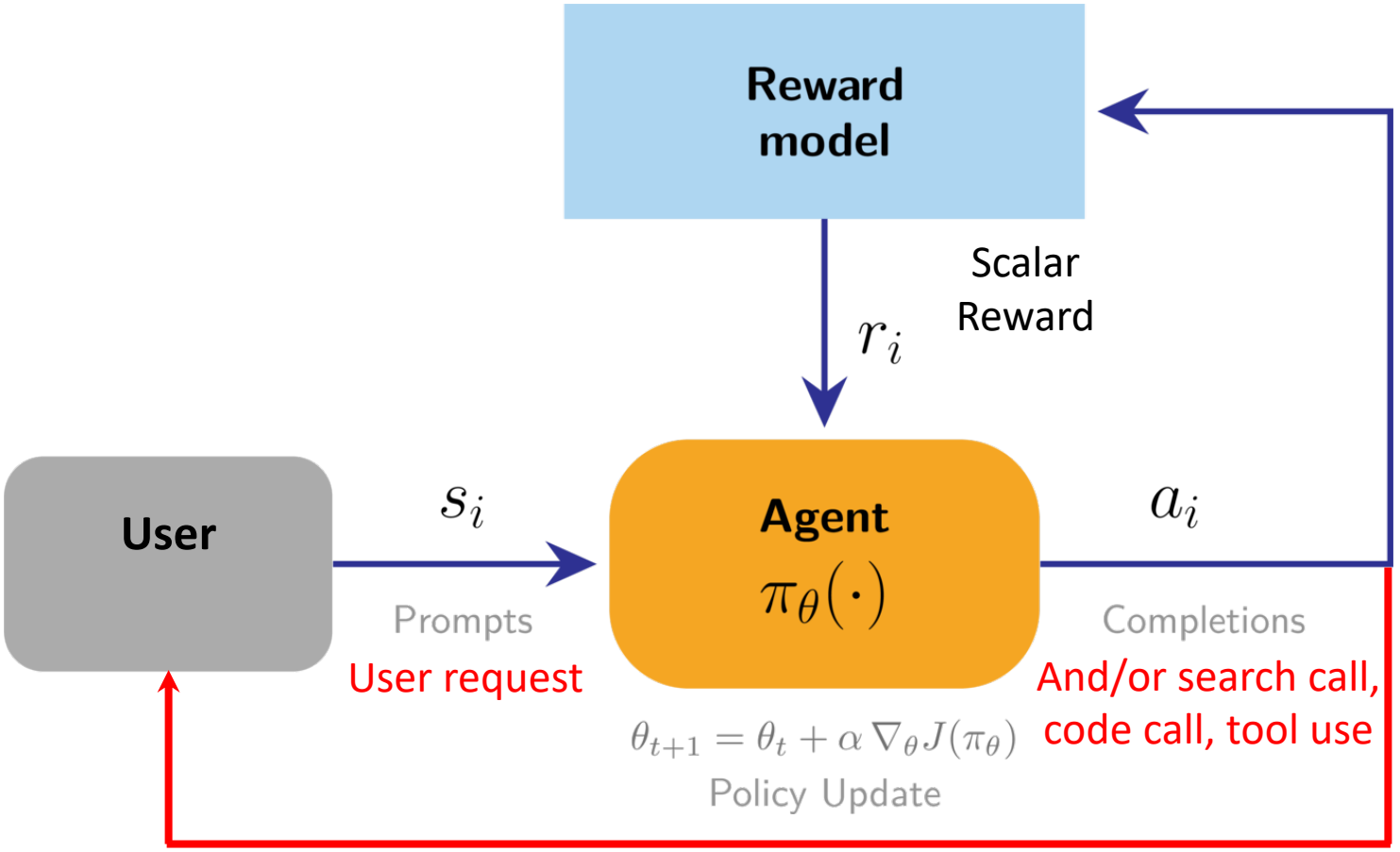
InstructGPT

```
The frog is a symbol of wisdom
He knows all the secrets of the world
He is a master of disguise
And a great teacher of life
He is a symbol of transformation
And the bringer of change
He is the frog who has seen it all
And knows the meaning of it all
```

**ChatGPT (2022)**

- Similar methodology
- More, bigger, better (limited specifics).
- More dialog-oriented training data.

# Agentic AI loop (multi-turn interactions)



# Recap

- Policy gradient methods offer a conceptually simple class of methods for reinforcement learning.
- They work by directly optimizing the policy (rather than the value function) by approximating the gradient of the value function.
- Policy gradient methods attempt to maximize the likelihood of good trajectories.
- The policy gradient theorem enables us to estimate the gradient through Monte Carlo trajectory samples (REINFORCE algorithm).
- Advantages: no Markovian assumption, often effective for continuous action space problems.
- Disadvantages: high variance and on-policy (limited sample efficiency).

# References

1. Matteo Pirotta. FAIR. Reinforcement Learning. 2019, Lecture 5.

2. Matteo Pirotta. Reinforcement Learning Summer School, 2019. Policy Search: Actor-Critic Methods.