CS 957, System-2 AI
Test-Time Scaling

Mahdieh Soleymani

April 2025

Sharif University of Technology

Test-time scaling approaches

- **Explicit** inference-time scaling
  - Sampling or search-based methods
- **Implicit** inference-time scaling
  - Training to generate longer chains
    - RL
    - RL + SFT

Search or Learning

Repeated sampling + verifier

or

Learning?

Data for Reasoning Tasks

- Data as a AI fossil fuel

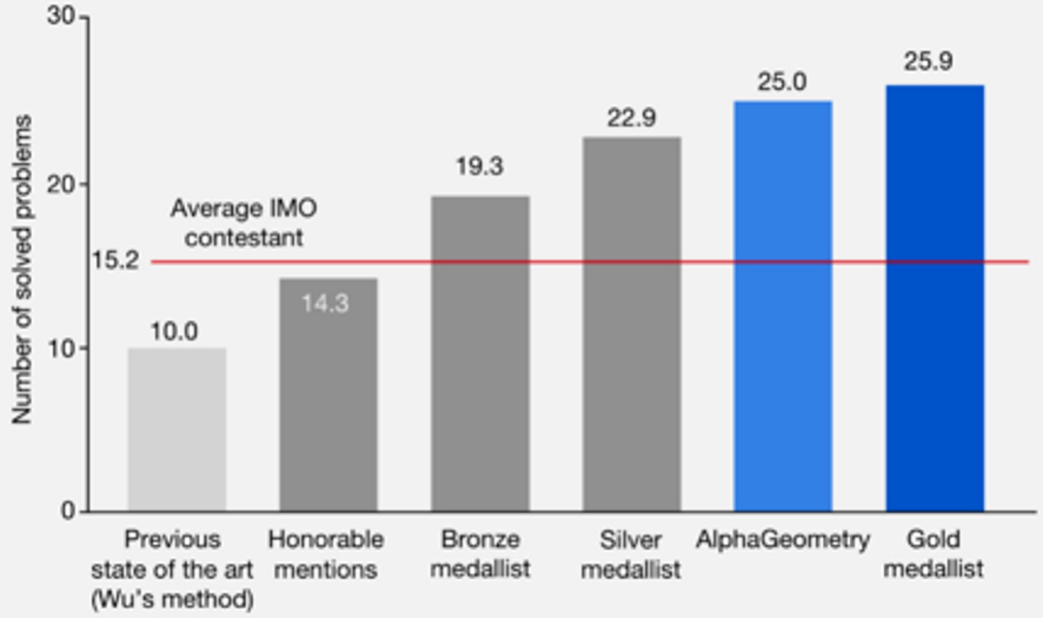
- Is it also true for reasoning problems??
  - reasoning problems often require drawing inferences about existing knowledge

- Which are other data sources for reasoning problems?
  - LLMs themselves also can be used as a source of generating reasoning data

AlphaGeometry

Solving olympiad geometry without human demonstrations

Synthesizing millions of theorems and proofs across different levels of complexity (using a symbolic deduction engine)



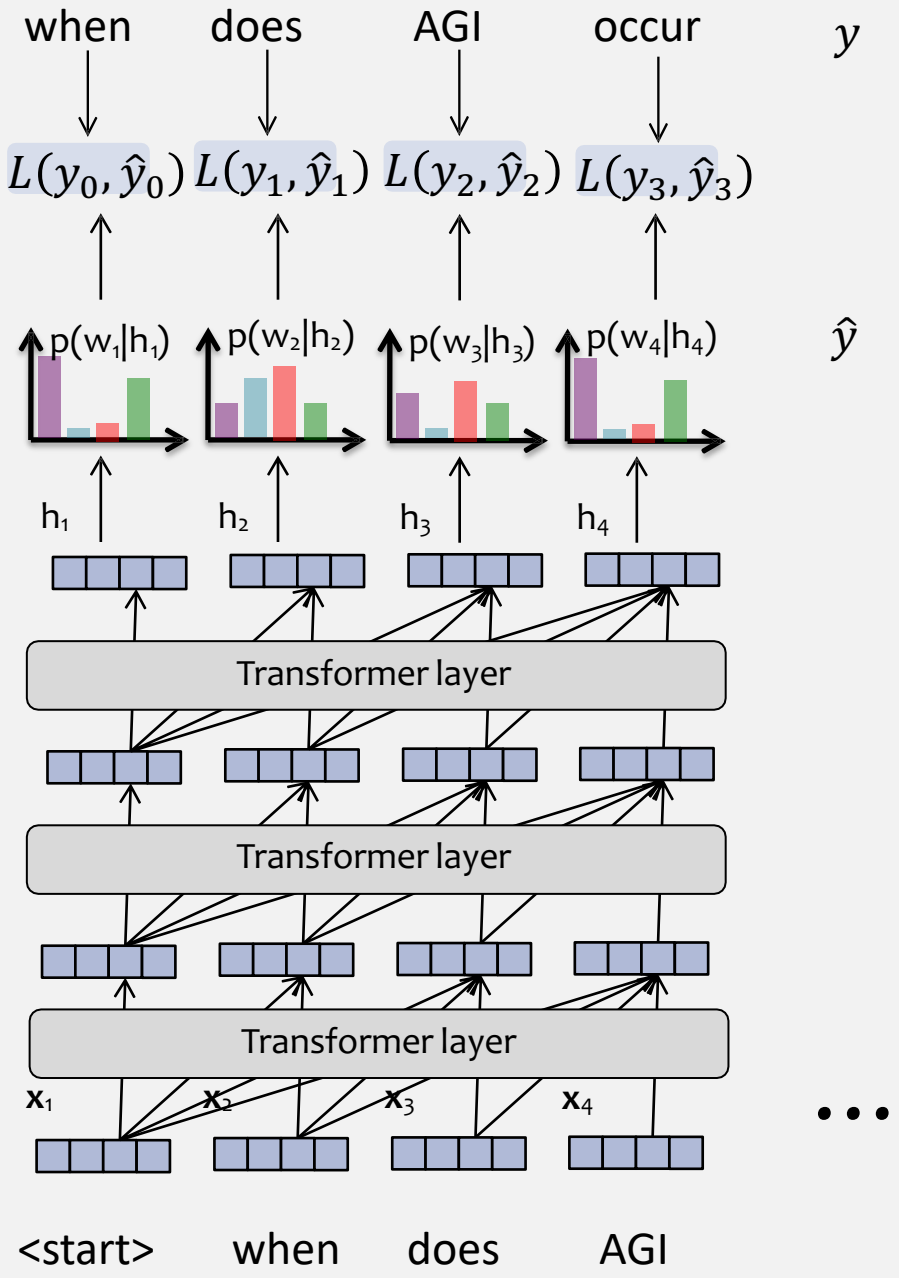
T.H. Trinh et al., Solving olympiad geometry without human demonstrations, Nature, 2024

Training data

- Sequences of <premise><conclusion><proof>
- Training an LM on synthesized data
- A language model effectively learns to generate the proof, conditioning on theorem premises and conclusion.

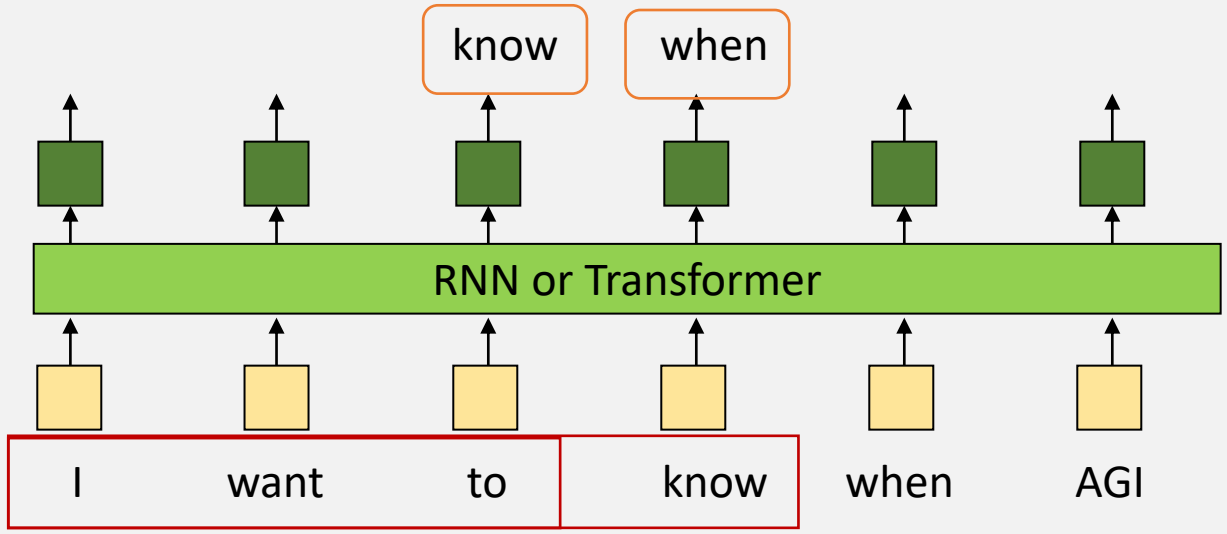
Next token Prediction

$$\max_{\theta} \sum_{n=1}^{N} \sum_{t=0}^{T} \log p_\theta(x_{t+1}^{(n)} | x_{1:t}^{(n)})$$



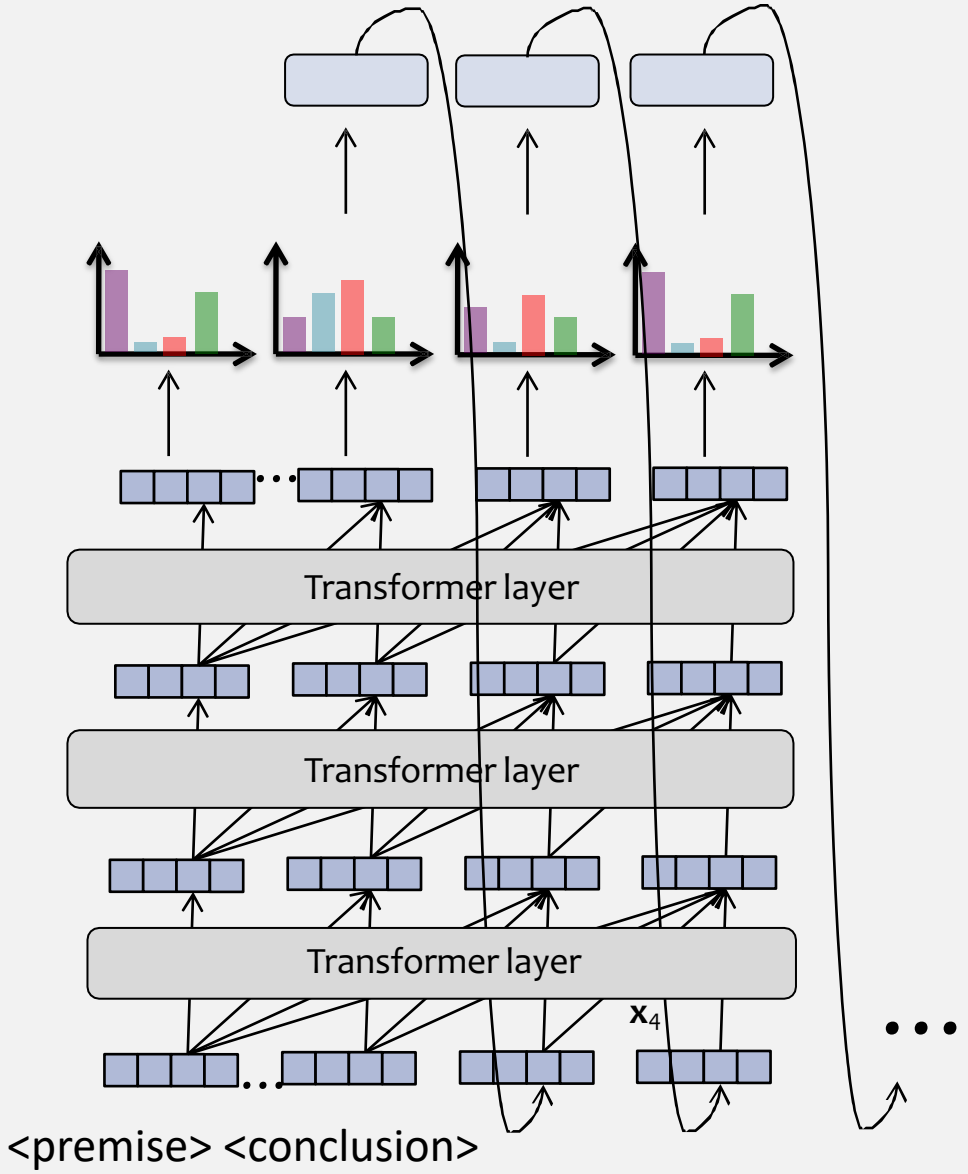
Language Modeling (LM)

Training time:



Next token prediction

Inference time



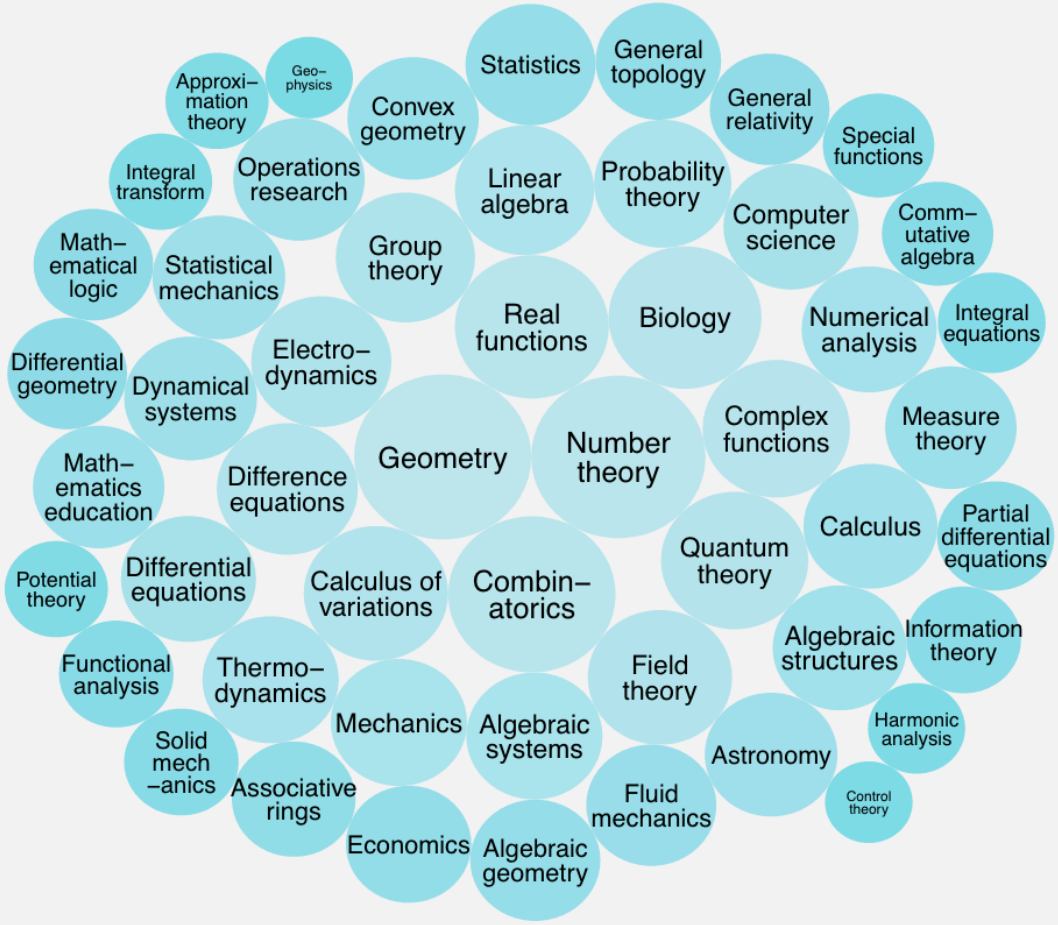
s1: Simple test-time scaling

Niklas Muennighoff [*1,3,4] **Zitong Yang** [*1] **Weijia Shi** [*2] **Xiang Lisa Li** [*1] **Li Fei-Fei** [1] **Hannaneh Hajishirzi** [2,3]
Luke Zettlemoyer [2] **Percy Liang** [1] **Emmanuel Candès** [1] **Tatsunori Hashimoto** [1]

Dataset

- 59,029 questions from 16 diverse sources
- For each question, a reasoning trace and solution using the Google Gemini Flash Thinking API are generated.
- Final selection of **1K samples** according to three guiding principles
  - **Quality**: inspect samples and ignore datasets with poor formatting
  - **Difficulty**: samples should be challenging and require significant reasoning effort
  - **Diversity**: should stem from different fields to cover different reasoning tasks.

Source	Description	#Samples	Avg. thinking length
NuminaMATH (LI et al., 2024)	Math problems from online websites	30660	4.1K
MATH (Hendrycks et al., 2021)	Math problems from competitions	11999	2.9K
OlympicArena (Huang et al., 2024a)	Astronomy, Biology, Chemistry, Computer Science, Geography, Math, and Physics olympiad questions	4250	3.2K
OmniMath (Gao et al., 2024a)	Math problems from competitions	4238	4.4K
AGIEval (Zhong et al., 2023; Ling et al., 2017; Hendrycks et al., 2021; Liu et al., 2020; Zhong et al., 2019; Wang et al., 2021)	English, Law, Logic and Math problems from the SAT, LSAT and other exams	2385	1.2K
xword	Crossword puzzles	999	0.7K
OlympiadBench (He et al., 2024)	Math and Physics olympiad questions	896	3.9K
AIME (1983-2021)	American Invitational Mathematics Examination	890	4.7K
TheoremQA (Chen et al., 2023)	Computer Science, Finance, Math, and Physics university-level questions relating to theorems	747	2.1K
USACO (Shi et al., 2024)	Code problems from the USA Computing Olympiad	519	3.6K
JEEBench (Arora et al., 2023)	Chemistry, Math, and Physics problems used in the university entrance examination of the Indian Institute of Technology	515	2.9K
GPQA (Rein et al., 2023)	PhD-Level Science Questions	348	2.9K
SciEval (Sun et al., 2024)	Biology, Chemistry, and Physics problems from various sources	227	0.7K
s1-prob	Stanford statistics qualifying exams	182	4.0K
LiveCodeBench (Jain et al., 2024)	Code problems from coding websites (LeetCode, AtCoder, and CodeForces)	151	3.5K
s1-teasers	Math brain-teasers crawled from the Internet	23	4.1K
All 59K questions	Composite of the above datasets with reasoning traces and solutions	59029	3.6K



we first choose one domain uniformly at random. Then, we sample one problem from this domain according to a distribution that favors longer reasoning traces (Difficulty). This process is repeated until we have 1,000 total samples

N. Muennighoff et al., s1: Simple test-time scaling, 2025

Training

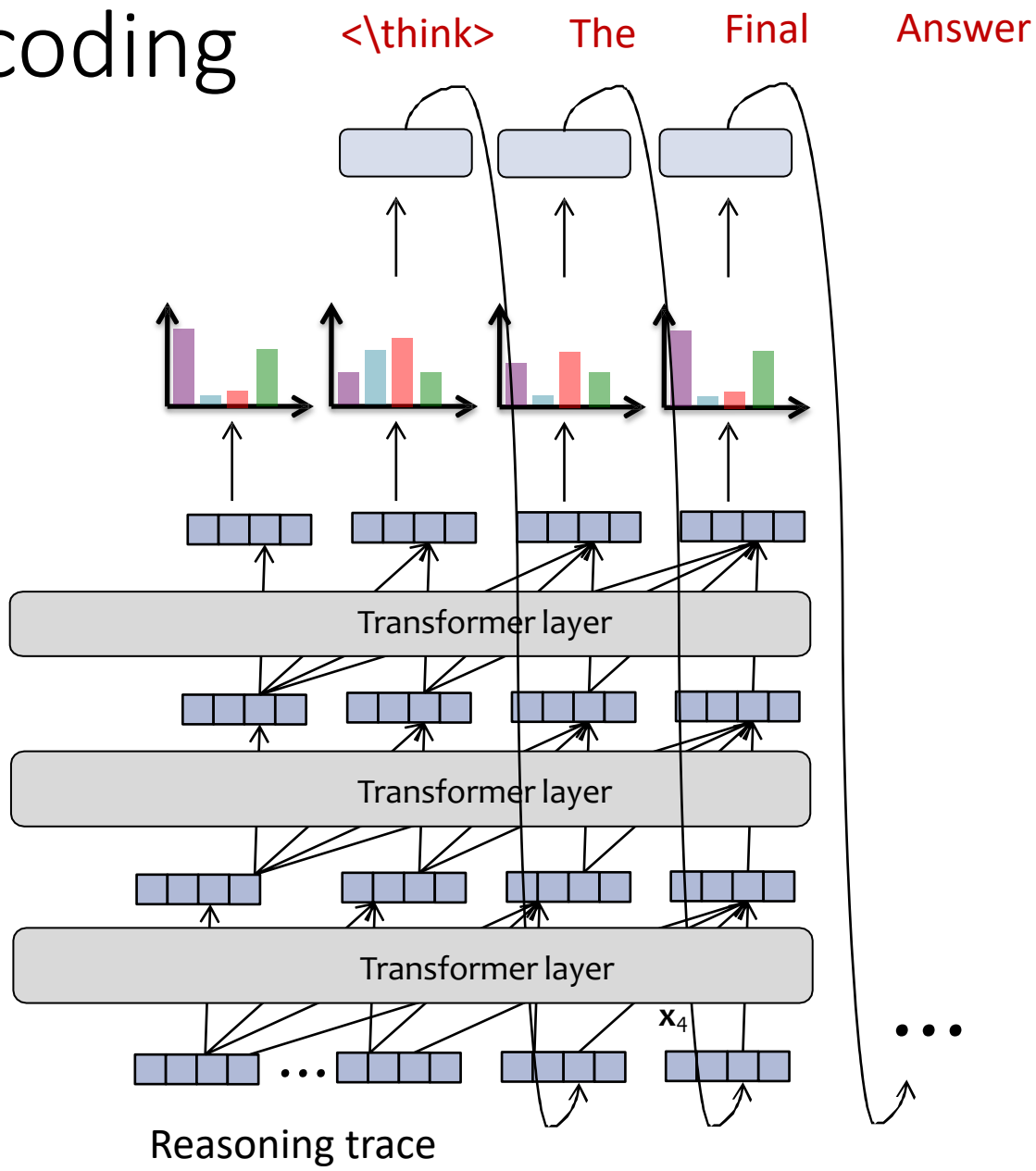
- Supervised fine-tuning (SFT) with next-token prediction
- Training Qwen2.532B-Instruct on 1,000 samples to produce s1-32B,

Test-time scaling

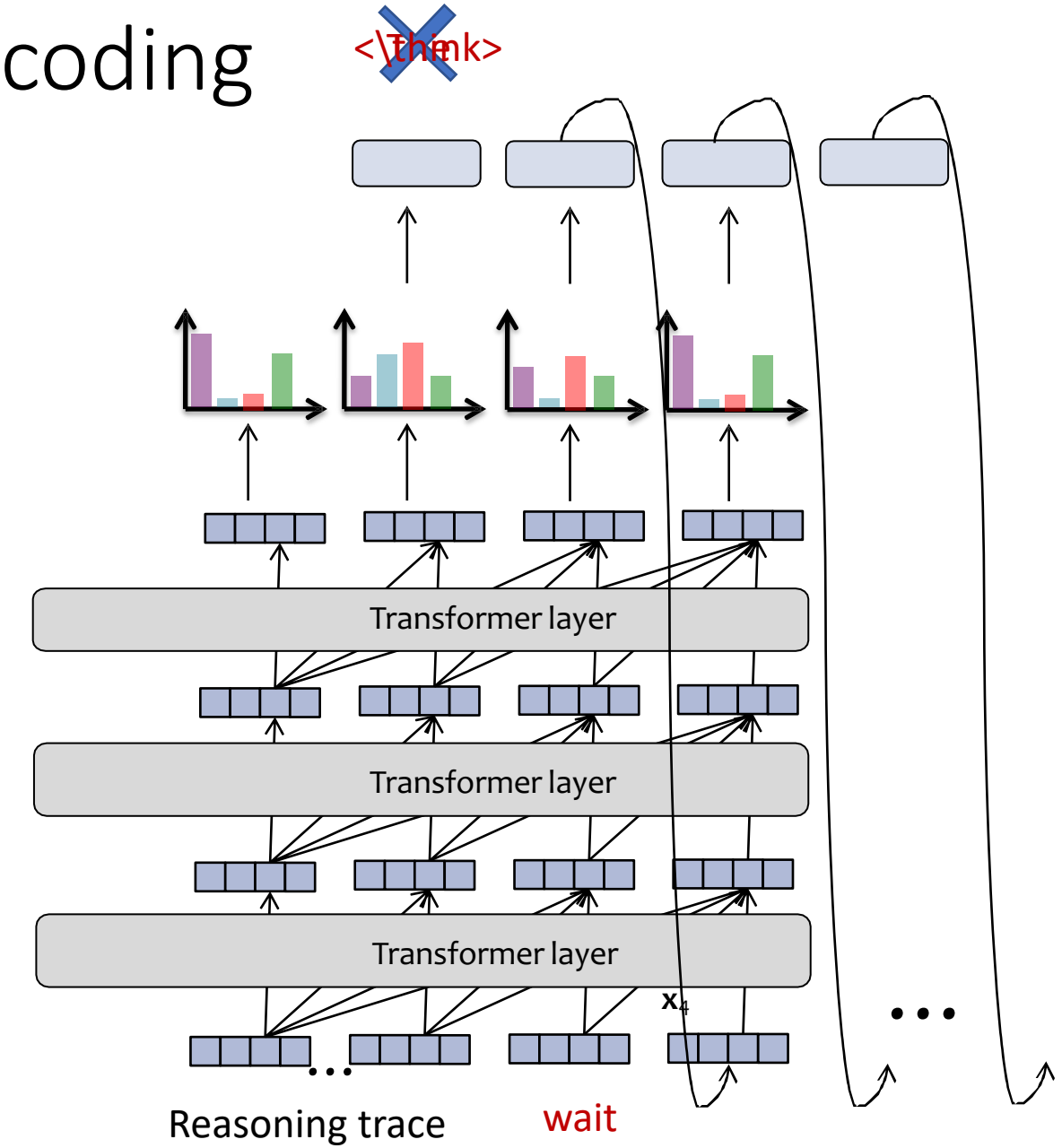
1) Sequential: later computations depend on earlier ones (e.g., a long reasoning trace)
2) Parallel: computations run independently (e.g., majority voting)

- Sequential scaling should scale better, since later computations can build on intermediate results, allowing for deeper reasoning and iterative refinement.

Manipulate the decoding process



Manipulate the decoding process



Test-time budget forcing

- controlling thinking duration leads to a strong reasoning model that scales in performance with more test-time compute.

- controls test-time compute by forcefully:

  - **terminating** the model's thinking process by simply appending the end-of-thinking token delimiter and "**Final Answer**:"

  - **lengthening** the thinking process by appending "**Wait**" multiple times to the model's generation when it tries to end.
    - This can lead the model to doublecheck its answer, often fixing incorrect reasoning steps.

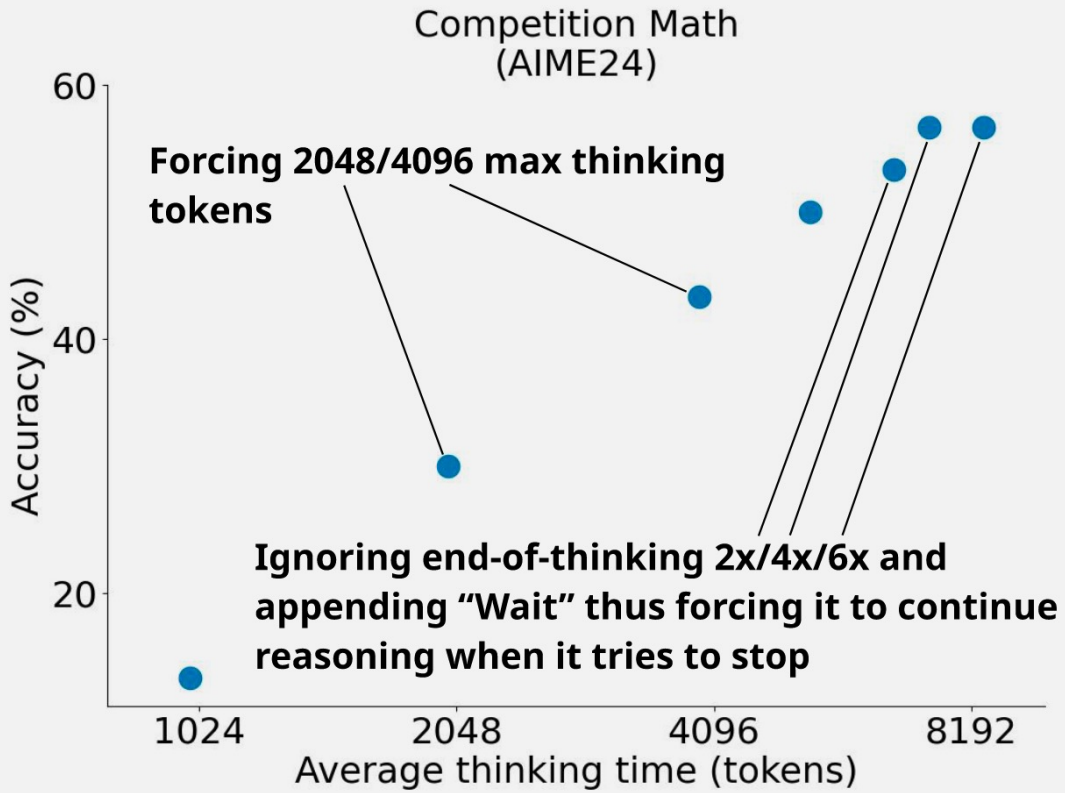
N. Muennighoff et al., s1: Simple test-time scaling, 2025v

How many r in raspberry? **Question**

Let's break down the process of counting the letter 'r' in the word "raspberry" ...
* First letter: 'r' - This is an 'r', count = 1.
* Second letter: 'a' - Not an 'r', count remains 1 ...
* Sixth letter: 'e' - Not an 'r', count remains 1.
* Seventh letter: 'r' - This is an 'r', count = 2.
* Eighth letter: 'y' - Not an 'r', count remains 2 ...
The number of 'r's in "raspberry" is 2.
Wait, let's re-read the question carefully. It asks "How many r in raspberry?" ... * r - a - s - p - b - e - r - r - y ... * First 'r' ... * Second 'r' ... * Third 'r' ... Count = 3 ... **Reasoning trace**

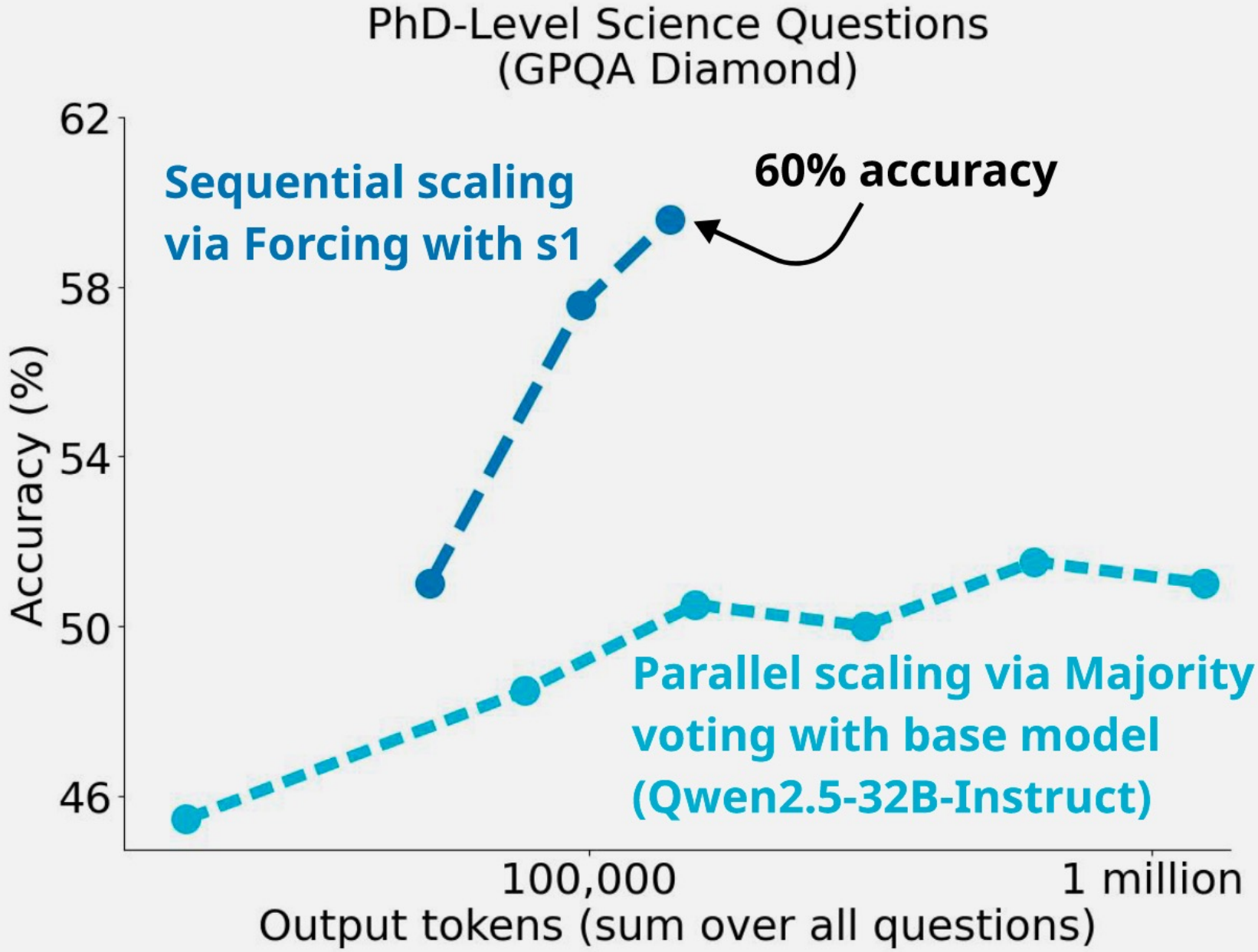
My initial answer of 2 was incorrect due to a quick reading of the word. **Final Answer:** The final answer is $\boxed{3}$ **Response**

N. Muennighoff et al., s1: Simple test-time scaling, 2025

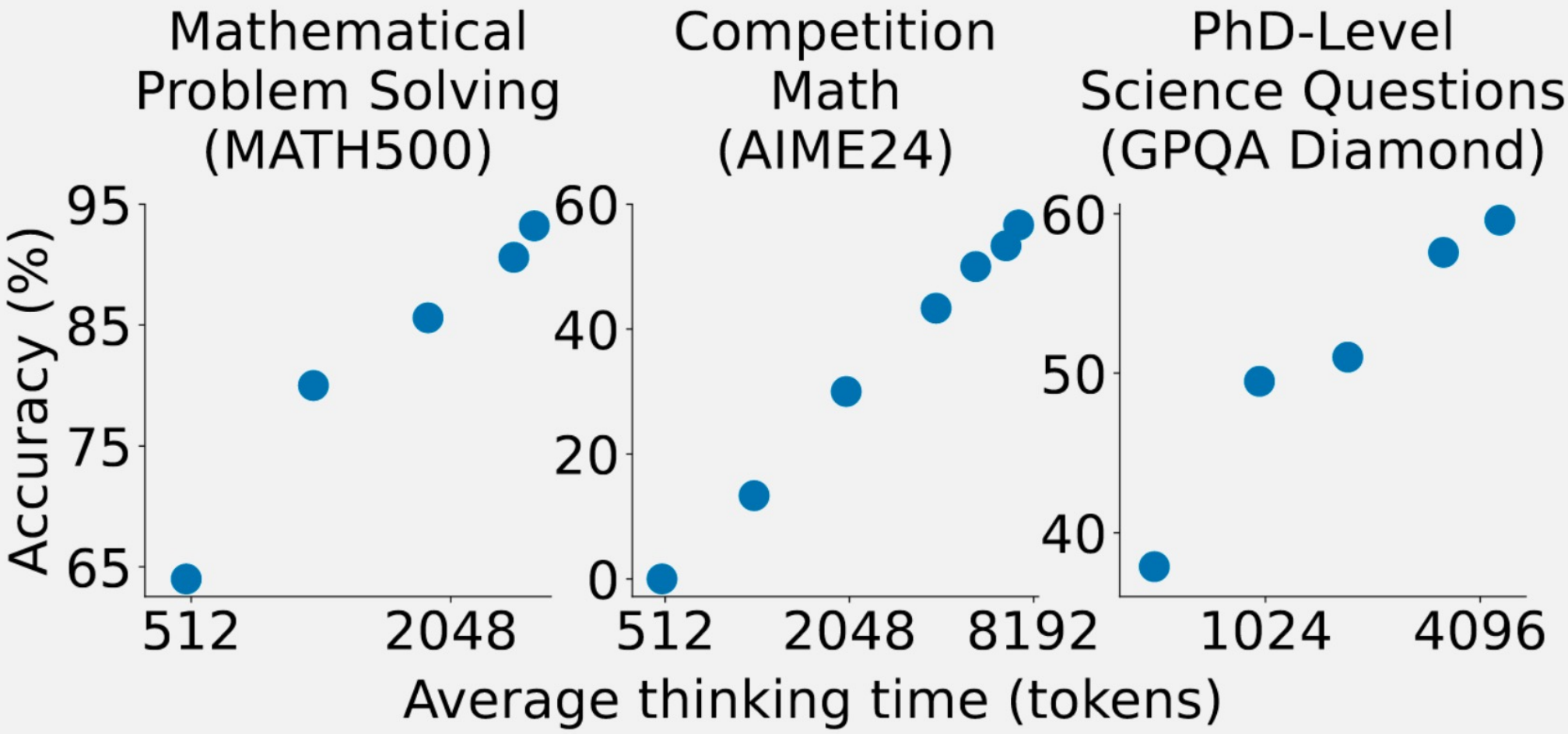


(a) Sequential scaling via budget forcing

N. Muennighoff et al., s1: Simple test-time scaling, 2025



(b) Parallel scaling via majority voting



N. Muennighoff et al., s1: Simple test-time scaling, 2025

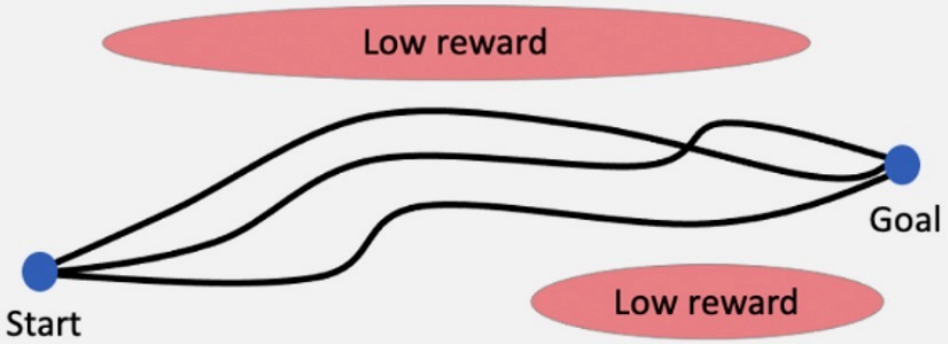
Model	AIME 2024	MATH 500	GPQA Diamond
1K-random	36.7 [-26.7%, -3.3%]	90.6 [-4.8%, 0.0%]	52.0 [-12.6%, 2.5%]
1K-diverse	26.7 [-40.0%, -10.0%]	91.2 [-4.0%, 0.2%]	54.6 [-10.1%, 5.1%]
1K-longest	33.3 [-36.7%, 0.0%]	90.4 [-5.0%, -0.2%]	59.6 [-5.1%, 10.1%]
59K-full	53.3 [-13.3%, 20.0%]	92.8 [-2.6%, 2.2%]	58.1 [-6.6%, 8.6%]
s1K	50.0	93.0	57.6

N. Muennighoff et al., s1: Simple test-time scaling, 2025

Post-training techniques

- Supervised Fine-Tuning (SFT)
- Reinforcement Learning (RL)

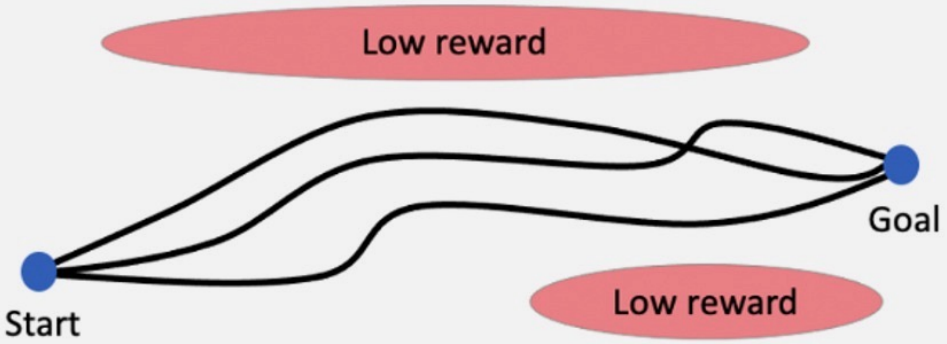
SFT



Exposure bias

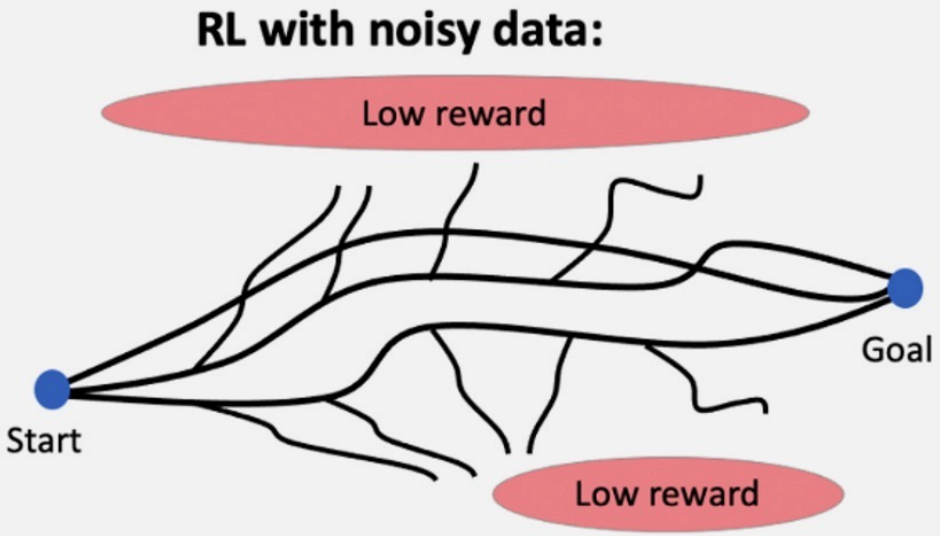
Adapted from https://bair.berkeley.edu/

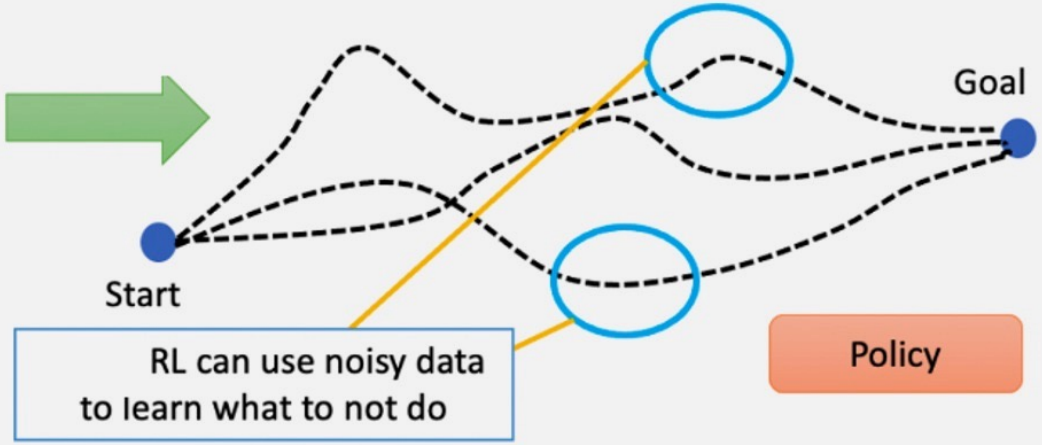
SFT



Exposure bias

RL





Adapted from https://bair.berkeley.edu/

LLMs & RL

- Scaling reinforcement learning (RL) unlocks a new axis for the continued improvement of artificial intelligence
- LLMs can scale their training data by learning to explore with rewards.

RL vs. SFT

SFT tends to memorize the training data

RL shows superior out-of-distribution generalization

SFT is still helpful for effective RL training

T. Chu et al., SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training, 2025

Reasoning (and planning) by LLMs

- LLMs + Explicit test time scaling

- LLMs + SFT

- **LLMs + Learning to explore:** RL as an approach making autonomous systems
  - LLMs themselves as a source of generating reasoning data

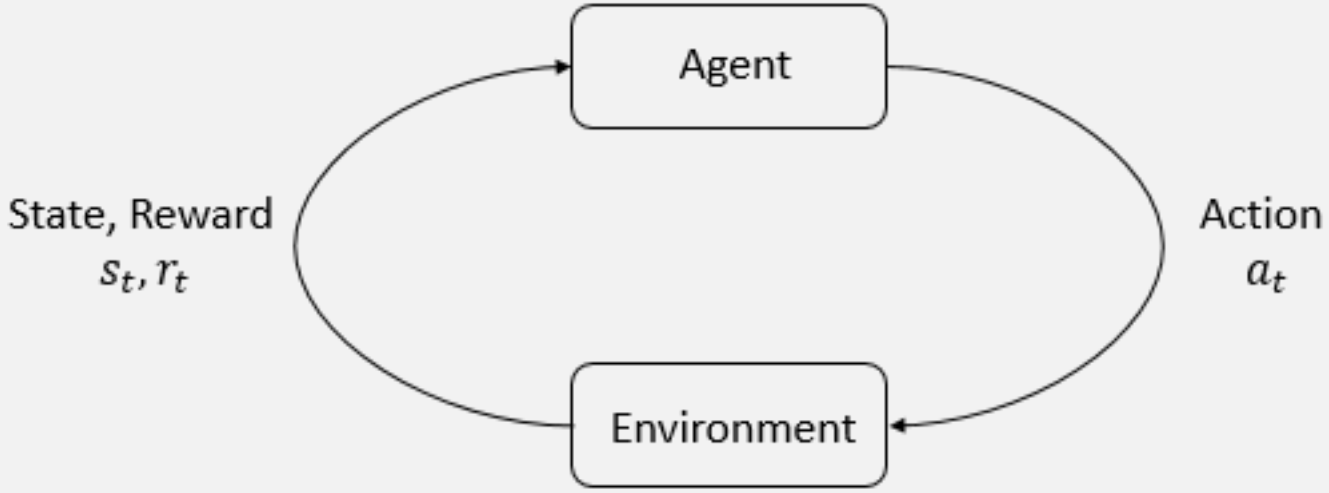
Is there another way to collect data?

LLMs themselves as a source for generating reasoning data

Reinforcement Learning (RL)

- The agent actively interacts with an environment.
  - may influence the environment that it operates in
  - Collects its data interactively

- Agent learns from its successes and mistakes



object classification

supervised learning

iid data

large labeled, curated dataset

well-defined notions of success

object manipulation

sequential decision making

action affects next state

how to collect data? what are the labels?

what does success mean?

Definitions

- S is the set of all valid states
- A is the set of all valid actions,
- $R: S \times A \times S \rightarrow \mathbb{R}$ is the reward function with $r_t = (s_t, a_t, s_{t+1})$
- $P: S \times A \rightarrow \mathcal{P}(S)$ is the transition probability function
  - $P(s'|s, a)$ is the probability of transitioning into s' if you are in s and take action a
- ρ_0 is the starting state distribution.

Definitions: Trajectory & Return

- A trajectory τ is a sequence of states and actions in the world

$$\tau = (s_0, a_0, s_1, a_1, \dots)$$

- Return: cumulative reward over a trajectory
  - **finite-horizon undiscounted return**

$$r(\tau) = \sum_{t=0}^{T} r_t = \sum_{t=0}^{T} r(s_t, a_t, s_{t+1})$$

  - **infinite-horizon discounted return** ($\gamma \in (0,1)$)

$$r(\tau) = \sum_{t=0}^{T} \gamma^t \, r_t$$

Definitions: Policy

- A **policy** is a rule to decide what actions to take.
  - **Deterministic**: $\pi: S \to A$
    - $a_t = \pi(s_t)$
  - **Stochastic**: $\pi: S \to \mathcal{P}(A)$
    - $a_t \sim \pi(. \mid s_t)$

RL Optimization Problem

- The central optimization problem in RL:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} E_{\tau \sim \pi}[r(\tau)]$$

Value Functions

$$\tau = (s_0, a_0, s_1, a_1, \dots)$$

$V^\pi(s)$: How good for the agent to be in the state s when its policy is π

$$V^\pi(s) = E_{\tau\sim\pi}[r(\tau)|s_0 = s]$$

Total reward starting from s and following policy π

$Q^\pi(s,a)$: How good for the agent to be in the state s and do action a when its policy is π

$$Q^\pi(s,a) = E_{\tau\sim\pi}[r(\tau)|s_0 = s, a_0 = a]$$

Total reward starting from s, taking a and then following policy π

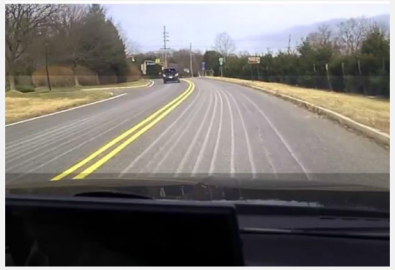
Policy Optimization

- Model-free RL

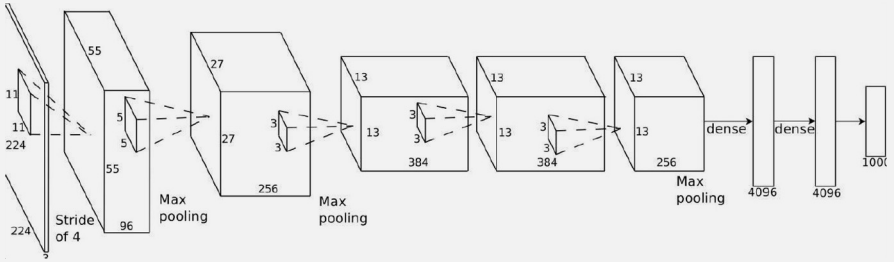
- **Policy Optimization**: directly optimize for the thing you want
  - The policy is explicitly modeled as $\pi_\theta(a|s)$ and optimize the parameters θ
  - This optimization is almost always performed on-policy
    - each update only uses data collected while acting according to the most recent policy.

- also usually involves learning an approximator $V_\phi(s)$ for the on-policy value function $V^\pi(s)$
  - used in figuring out how to update the policy.

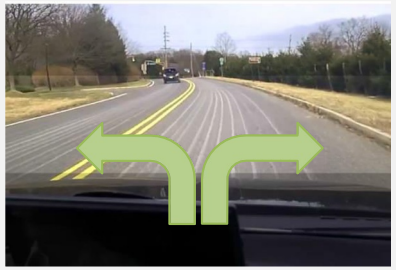
Imitation Learning



$\mathbf{s}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

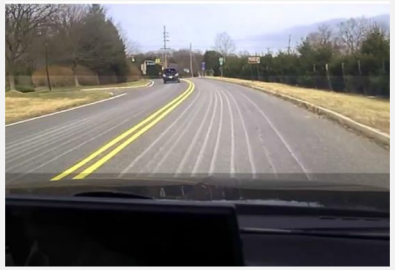


$\mathbf{a}_t$

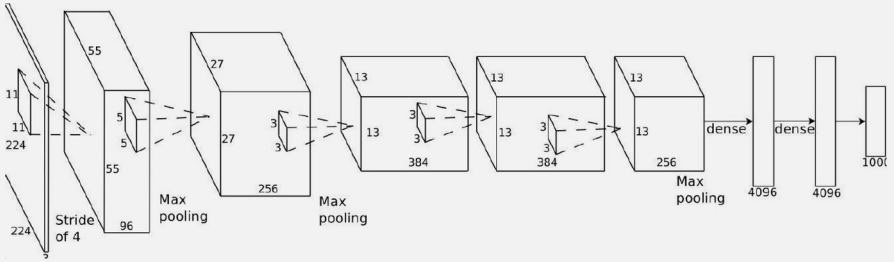
Images: Bojarski et al. ‘16, NVIDIA

Slide adapted from Sergey Levine

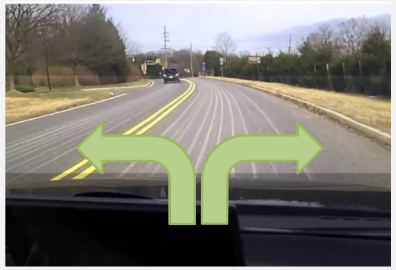
Imitation Learning



$\mathbf{s}_t$

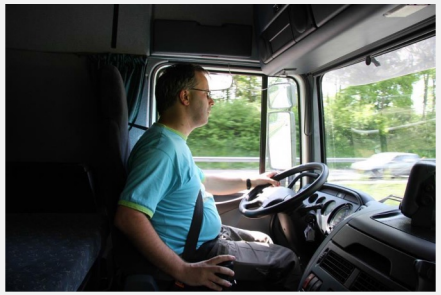


$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$



$\mathbf{a}_t$

Imitation Learning vs Reinforcement Learning?



$\mathbf{s}_t$

$\mathbf{a}_t$

training data

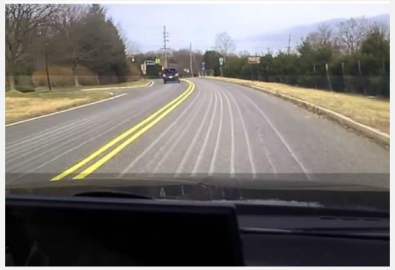
supervised learning

$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

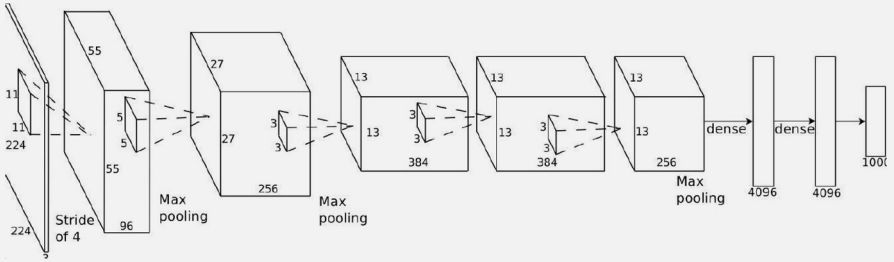
Images: Bojarski et al. ‘16, NVIDIA

Slide adapted from Sergey Levine

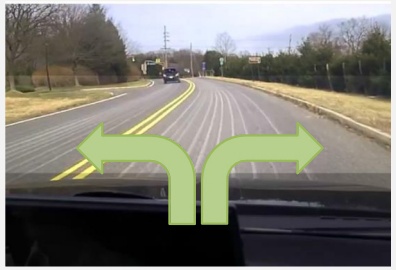
Reward functions



$\mathbf{s}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$



$\mathbf{a}_t$

which action is better or worse?

$r(\mathbf{s}, \mathbf{a})$: reward function

tells us which states and actions are better

high reward

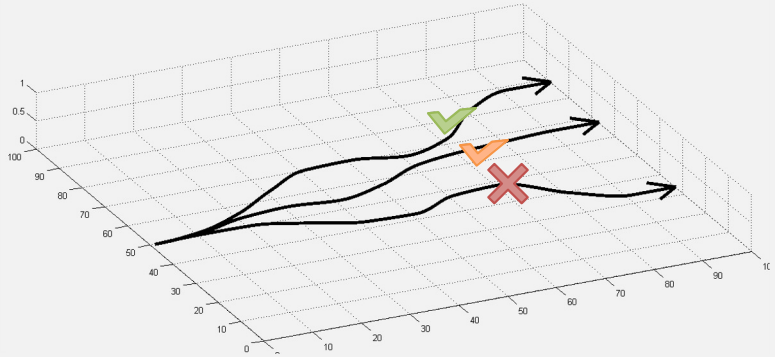
low reward

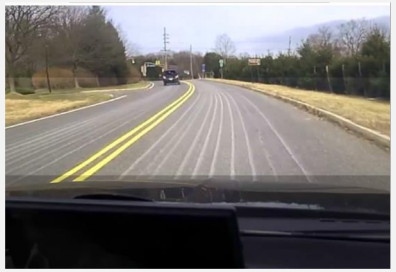
Slide adapted from Sergey Levine

Comparison to maximum likelihood

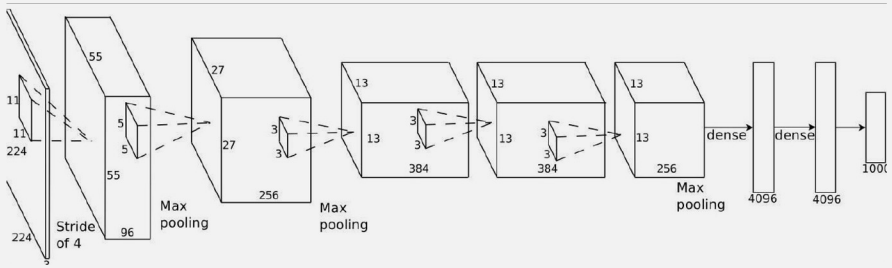
$$\text{policy gradient:} \quad \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

$$\text{maximum likelihood:} \quad \nabla_\theta J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right)$$

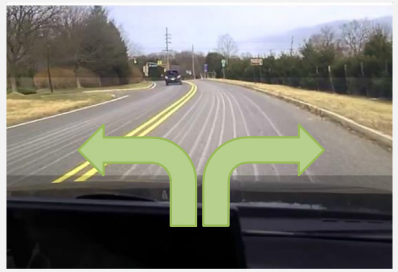




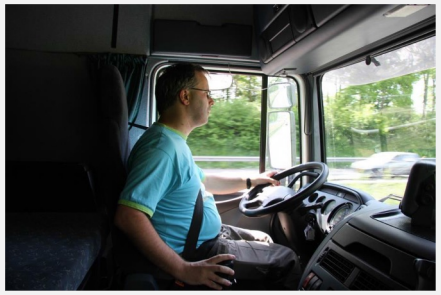
$\mathbf{s}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

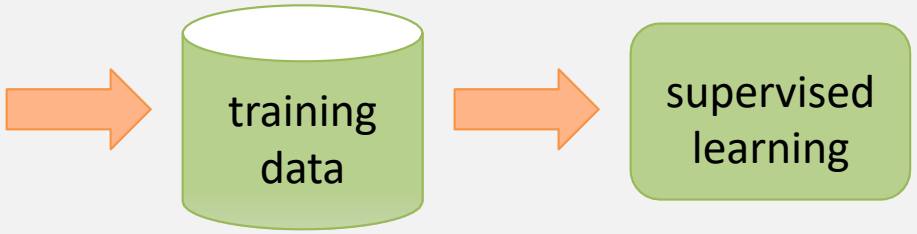


$\mathbf{a}_t$



$\mathbf{s}_t$

$\mathbf{a}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

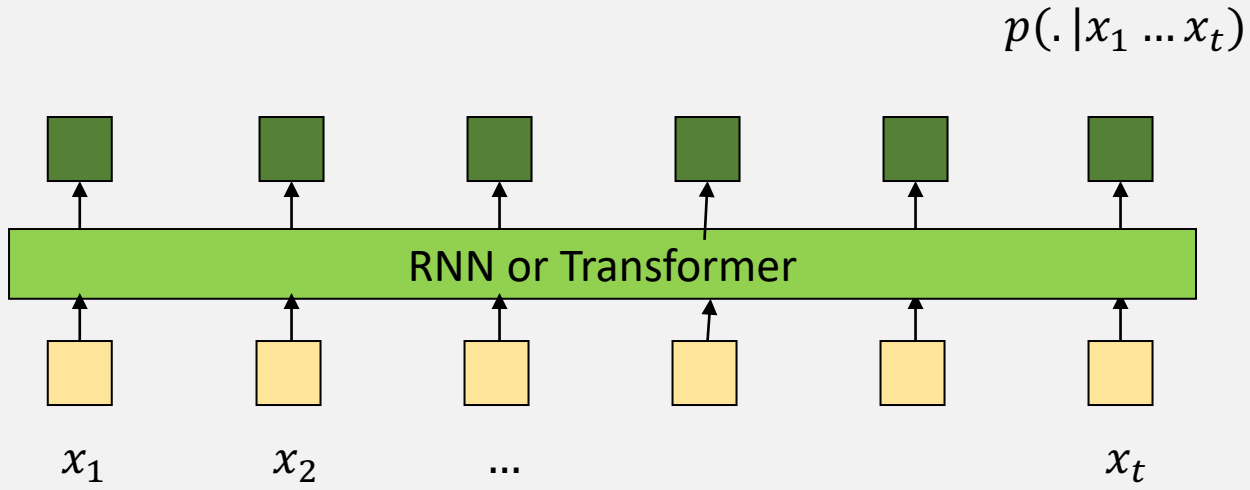
Slide adapted from Sergey Levine

LLM distribution as a policy

- We use an LM as a policy, which maps the problem statement and steps-so-far to a next step.

- In the RL formalism, this treats each step (e.g. generating a token) as an action, and the observation is provided by all the tokens so far.

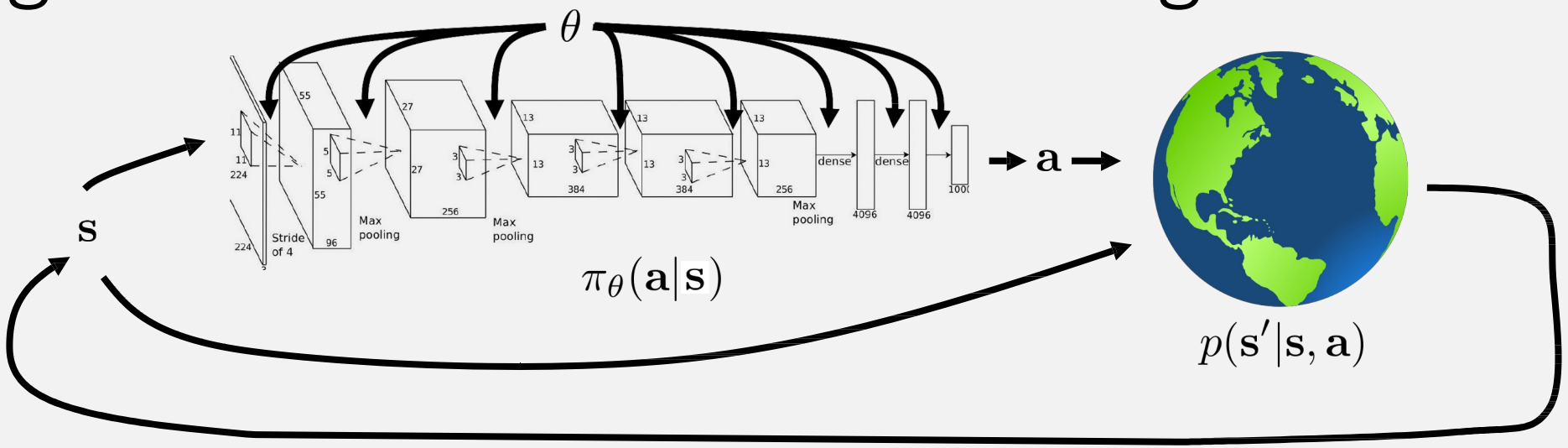
LLMs distribution as a policy

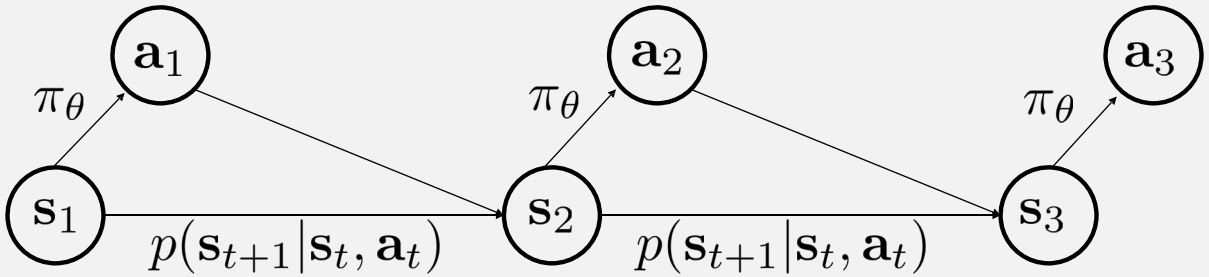


- $x_{t+1} \sim p(.|x_1, \dots, x_t)$

- State: $s_t = x_1, \dots, x_t$
- Action: $a_t \sim \pi(.|s_t) = p(.|x_1, \dots, x_t)$
- Transition function: $s_{t+1} = f(s_t, a_t) = x_1, \dots, x_t, x_{t+1}$

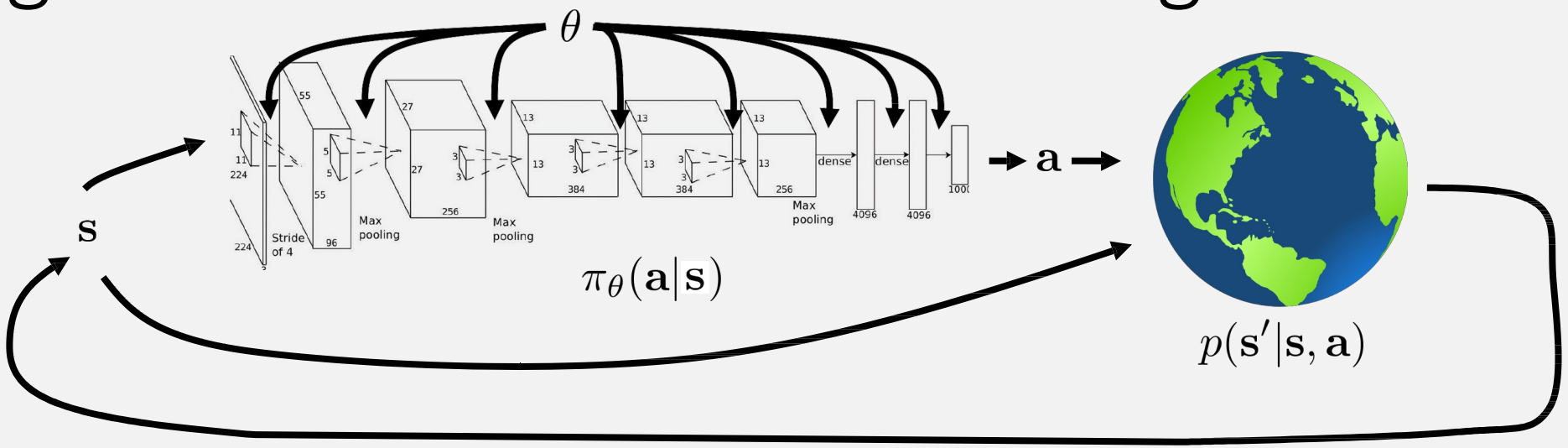
The goal of reinforcement learning

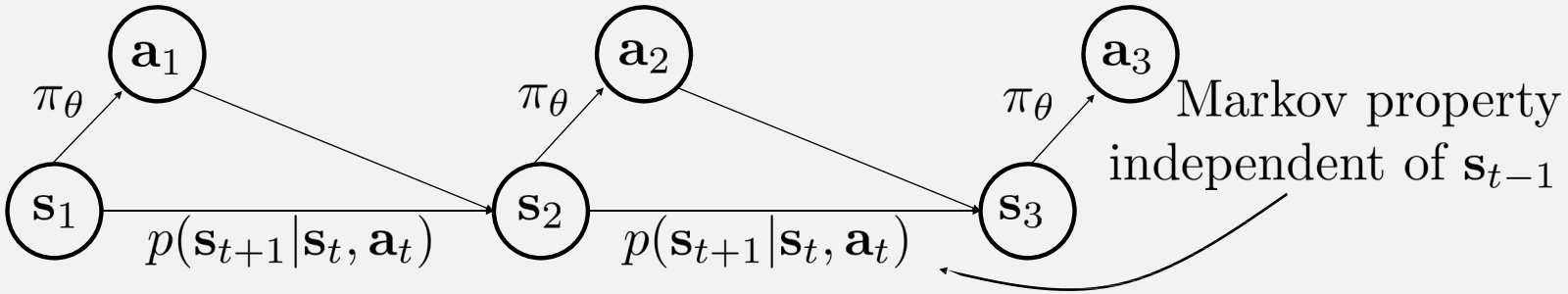




Slide adapted from Sergey Levine

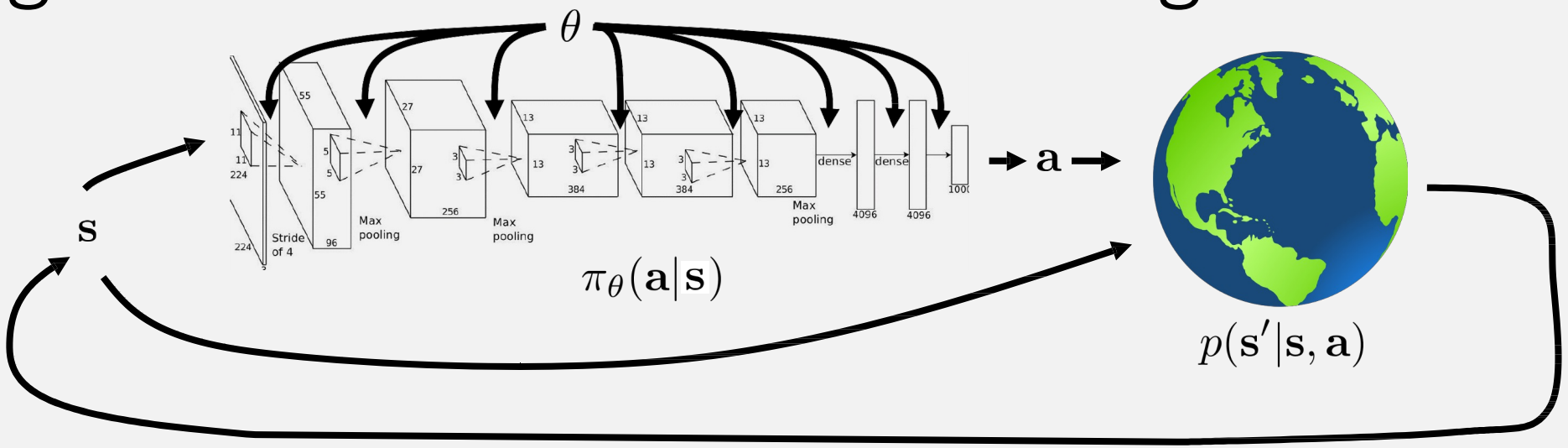
The goal of reinforcement learning

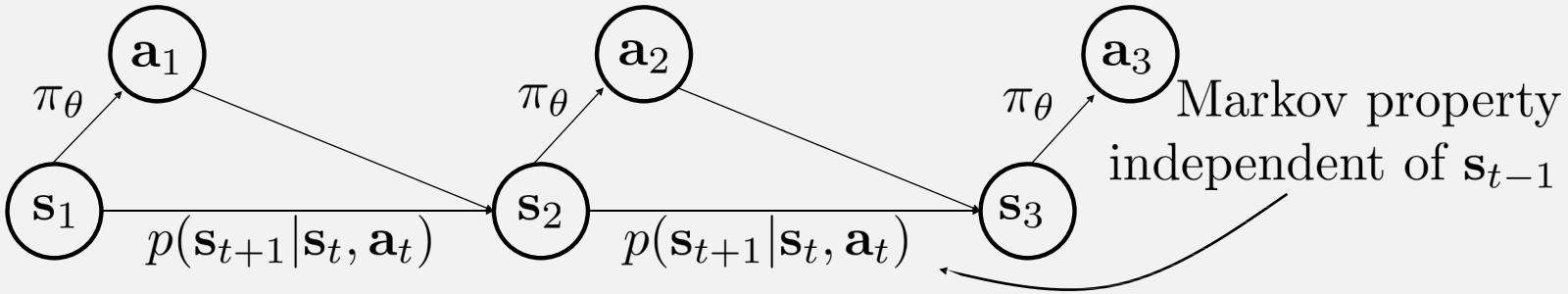




Slide adapted from Sergey Levine

The goal of reinforcement learning





$$\underbrace{\pi_\theta(\mathbf{s}_1,\mathbf{a}_1,\ldots,\mathbf{s}_T,\mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)$$

$$\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t,\mathbf{a}_t)\right]$$

Slide adapted from Sergey Levine

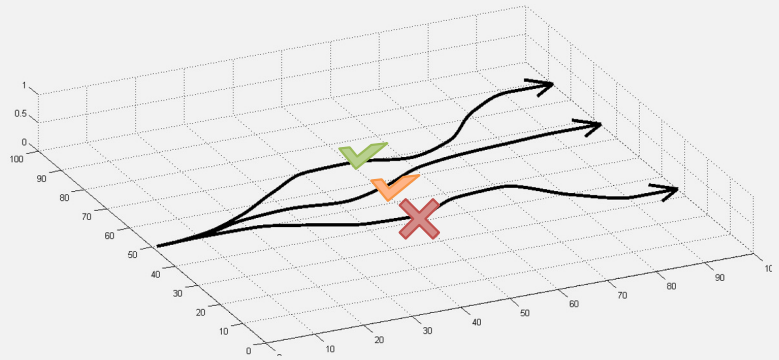
The optimal policy π^*

We want to find **optimal policy π^*** that maximizes the expected sum of rewards.
How do we handle the randomness (initial state, transition probability…)?

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi\right]$$
$$s_0 \sim p(s_0)$$
$$a_t \sim \pi(\cdot|s_t)$$
$$s_{t+1} \sim p(\cdot|s_t, a_t)$$

Evaluating the objective

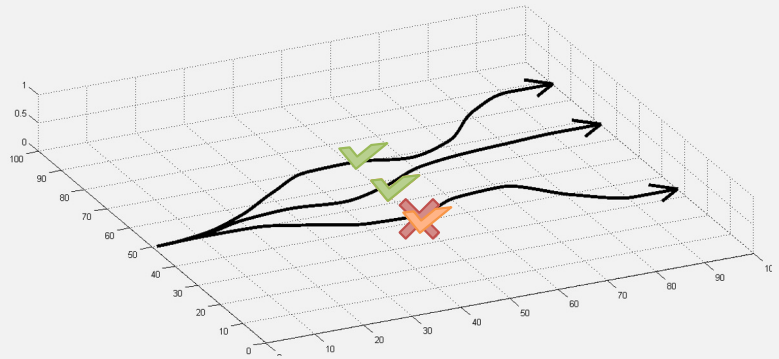
$$\theta^\star = \arg\max_\theta \underbrace{E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]}_{J(\theta)}$$



Slide adapted from Sergey Levine

Evaluating the objective

$$\theta^\star = \arg\max_\theta \underbrace{E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]}_{J(\theta)}$$



$$J(\theta) = E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right] \approx \frac{1}{N}\sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

sum over samples from π_θ

Slide adapted from Sergey Levine

Direct policy differentiation

$$\theta^\star = \arg\max_\theta \underbrace{E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]}_{J(\theta)}$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[\underbrace{r(\tau)}]$$

$$\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)$$

Slide adapted from Sergey Levine

Direct policy differentiation

$$\theta^\star = \arg\max_\theta \underbrace{E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]}_{J(\theta)}$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[\underbrace{r(\tau)}_{\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)}] = \int \pi_\theta(\tau) r(\tau) d\tau$$

Slide adapted from Sergey Levine

Direct policy differentiation

$$\theta^\star = \arg\max_\theta \underbrace{E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]}_{J(\theta)}$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[\underbrace{r(\tau)}_{\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)}] = \int \pi_\theta(\tau) r(\tau) d\tau$$

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau$$

Slide adapted from Sergey Levine

Direct policy differentiation

$$\theta^\star = \arg\max_\theta \underbrace{E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]}_{J(\theta)}$$

a convenient identity

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau)\frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau)$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[\underbrace{r(\tau)}_{\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)}] = \int \pi_\theta(\tau) r(\tau) d\tau$$

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau = \int \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau = E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

Slide adapted from Sergey Levine

Direct policy differentiation

$$\theta^\star = \arg\max_\theta J(\theta)$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[r(\tau)]$$

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

Slide adapted from Sergey Levine

Direct policy differentiation

$$\theta^\star = \arg\max_\theta J(\theta)$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[r(\tau)]$$

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

$$\underbrace{\pi_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

Slide adapted from Sergey Levine

Direct policy differentiation

$$\theta^\star = \arg\max_\theta J(\theta)$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[r(\tau)]$$

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

$$\underbrace{\pi_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

log of both sides

$$\log \pi_\theta(\tau) = \log p(\mathbf{s}_1) + \sum_{t=1}^{T}\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

Slide adapted from Sergey Levine

Direct policy differentiation

$$\theta^\star = \arg\max_\theta J(\theta)$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[r(\tau)]$$

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

$$\nabla_\theta \left[\log p(\mathbf{s}_1) + \sum_{t=1}^{T} \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\right]$$

$$\underbrace{\pi_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

log of both sides

$$\log \pi_\theta(\tau) = \log p(\mathbf{s}_1) + \sum_{t=1}^{T} \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

Slide adapted from Sergey Levine

Direct policy differentiation

$$\theta^\star = \arg\max_\theta J(\theta)$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[r(\tau)]$$

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

$$\pi_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T) = p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

$\pi_\theta(\tau)$

log of both sides

$$\log \pi_\theta(\tau) = \log p(\mathbf{s}_1) + \sum_{t=1}^{T}\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

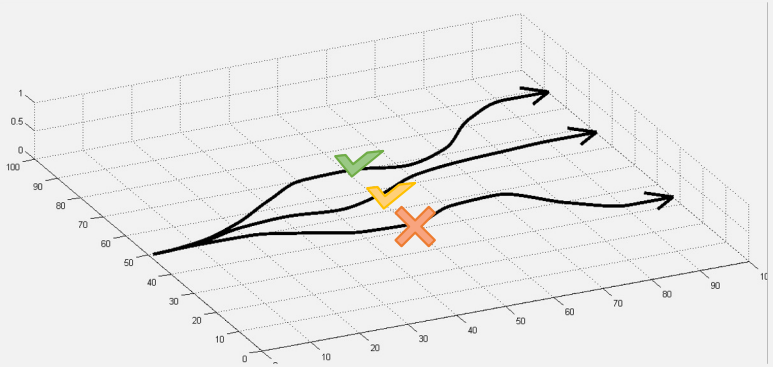
$$\nabla_\theta \left[\log p(\mathbf{s}_1) + \sum_{t=1}^{T}\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\right]$$

Dynamics model is not required

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right)\right]$$

Slide adapted from Sergey Levine

Evaluating the policy gradient

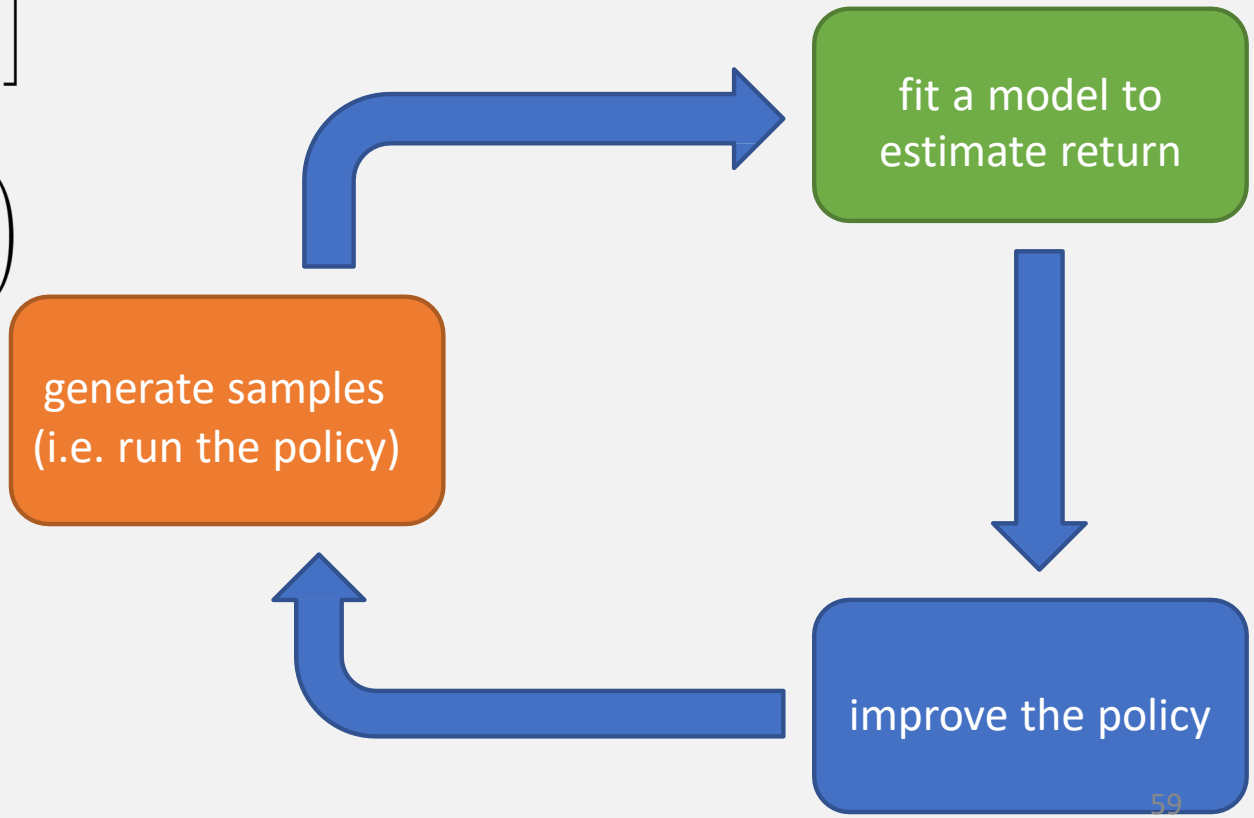


$$\text{recall: } J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

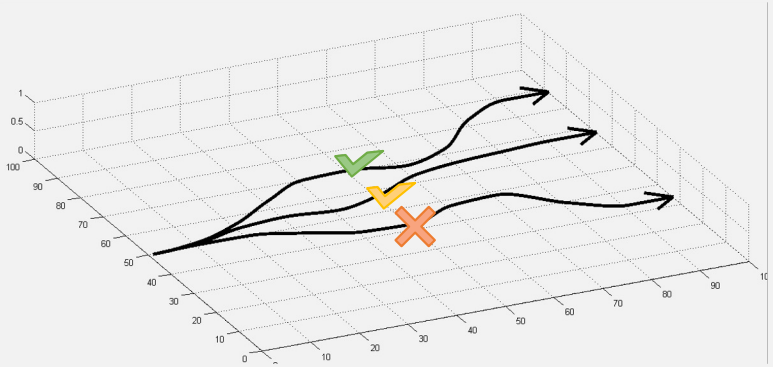
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

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



Slide adapted from Sergey Levine

Evaluating the policy gradient



$$\text{recall: } J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

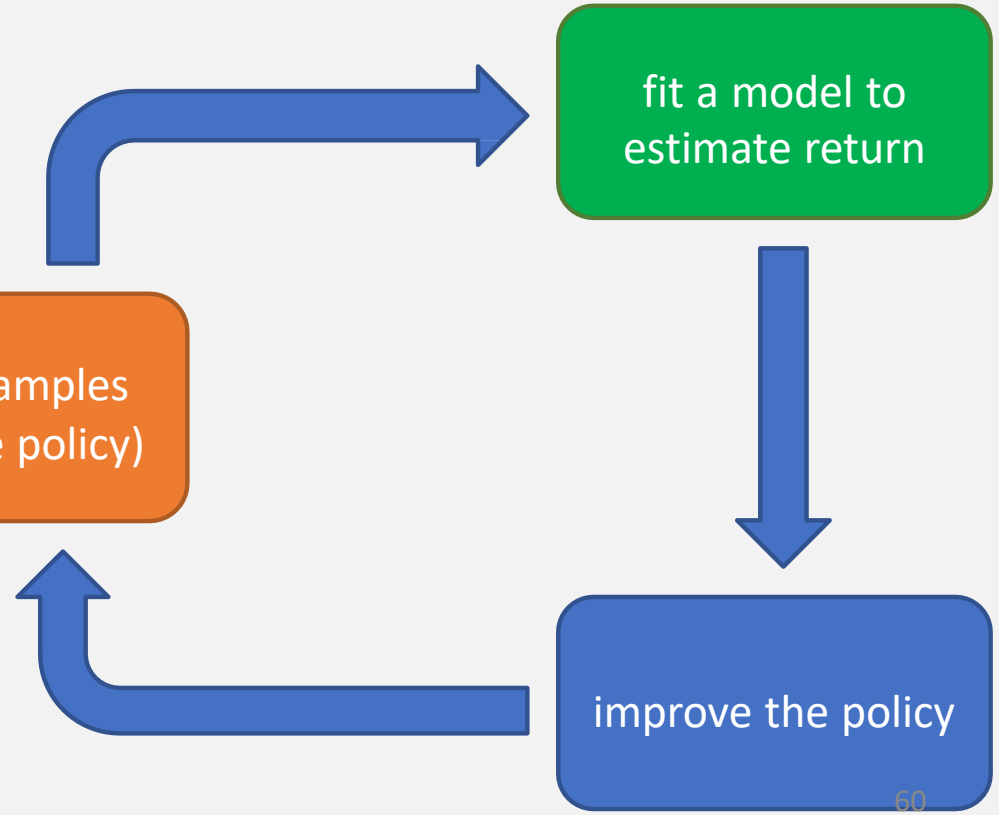
$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

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

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

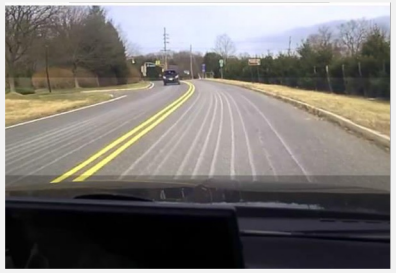


Slide adapted from Sergey Levine

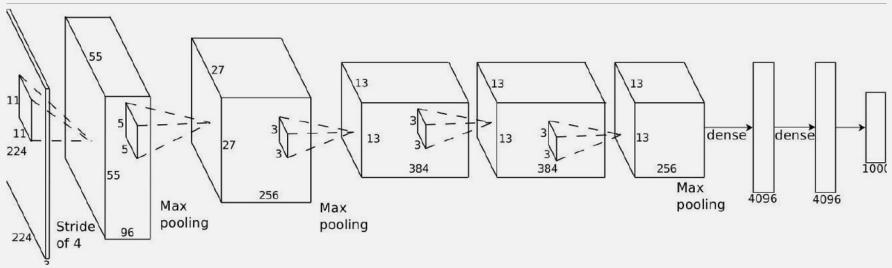
Comparison to maximum likelihood

$$\text{policy gradient:} \quad \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

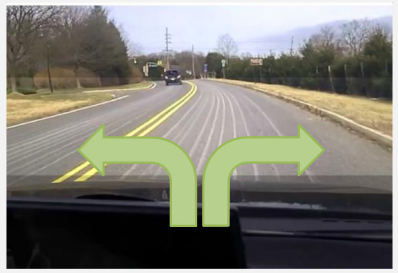
$$\text{maximum likelihood:} \quad \nabla_\theta J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right)$$



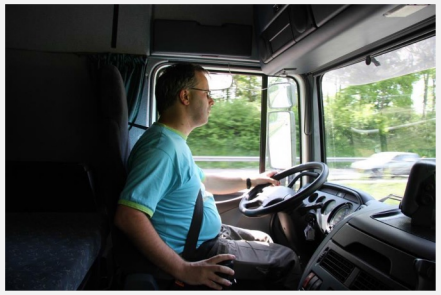
$\mathbf{s}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$



$\mathbf{a}_t$



$\mathbf{s}_t$

$\mathbf{a}_t$

training data

supervised learning

$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

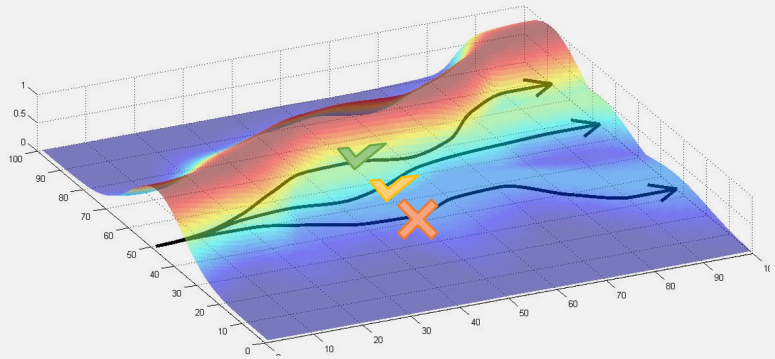
Slide adapted from Sergey Levine

What did we just do?

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \underbrace{\nabla_\theta \log \pi_\theta(\tau_i) r(\tau_i)}_{\sum_{t=1}^{T} \nabla_\theta \log_\theta \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}$$

maximum likelihood: $\nabla_\theta J_{\mathrm{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau_i)$



REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Slide adapted from Sergey Levine

What did we just do?

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

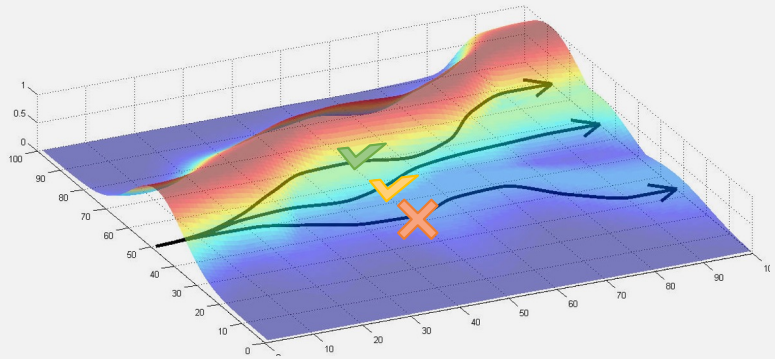
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \underbrace{\nabla_\theta \log \pi_\theta(\tau_i)}_{\sum_{t=1}^{T} \nabla_\theta \log_\theta \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} r(\tau_i)$$

maximum likelihood: $\nabla_\theta J_{\mathrm{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau_i)$

good stuff is made more likely

bad stuff is made less likely

simply formalizes the notion of “trial and error”!



REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Slide adapted from Sergey Levine

Gradient estimate

$$\nabla_\theta J(\theta) = E[r(\tau)\nabla_\theta \log \pi_\theta(\tau)]$$

$$\hat{g} = \frac{1}{N}\sum_{i=1}^{N} r(\tau_i)\nabla_\theta \log \pi_\theta(\tau_i)$$

It is unbiased: $\mathbb{E}[\hat{g}] = \nabla_\theta J(\theta)$

But, it has high variance

Policy Gradients

$$\text{policy gradient:} \quad \nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

Pros:

+ Simple

Cons:

- Produces a high-variance gradient

  Can be mitigated with baselines (used by all algorithms in practice), trust regions

- Requires on-policy data
- Cannot reuse existing experience to estimate the gradient!

  Importance weights can help, but also high variance

Reducing noise (or variance) of the gradient

- Causality and discount
- Baseline
- PPO

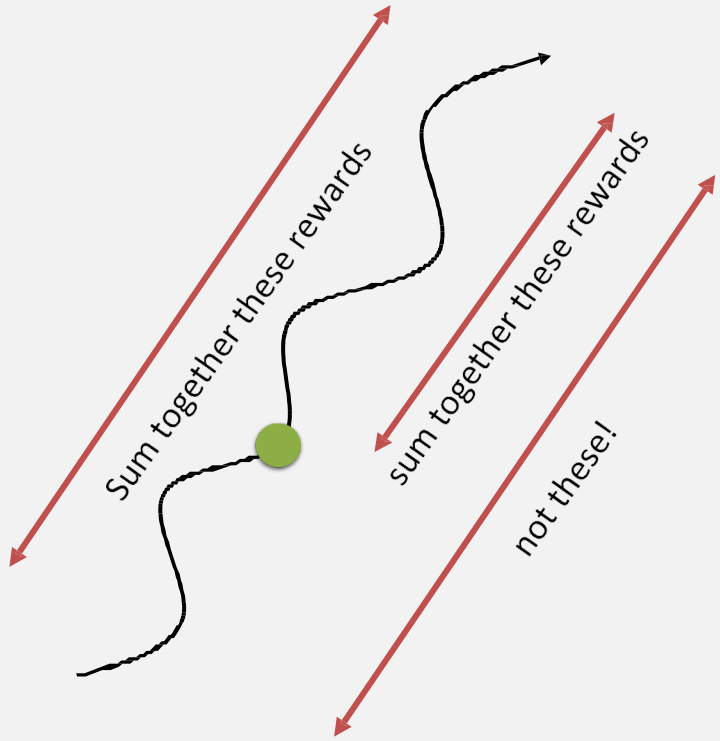
Causality

policy gradient: $$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)$$

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\left(\sum_{t'=1}^{T} r(\mathbf{a}_{i,t'},\mathbf{s}_{i,t'})\right)$$

Causality: $$\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\left(\sum_{t'=t}^{T} r(\mathbf{a}_{i,t'},\mathbf{s}_{i,t'})\right)$$

Reward “to go”



Discount factor

Gradient estimator:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta\left(a_{i,t}|s_{i,t}\right)\sum_{t=1}^{T} r\left(s_{i,t},a_{i,t}\right)$$

Use discount factor (a long with causality) to ignore rewards with delay:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta\left(a_{i,t}|s_{i,t}\right)\sum_{t'=t}^{T} \gamma^{t'-t} r\left(s_{i,t'},a_{i,t'}\right)$$

Baseline

Problem: The raw return isn't necessarily meaningful.
e.g., if rewards are all positive, you keep pushing up probabilities of actions.

We can use whether a reward is better or worse than what you expect to get

Subtract a baseline function dependent on the state:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta\left(a_{i,t}|s_{i,t}\right)\left(\sum_{t'=t}^{T} r\left(s_{i,t'}, a_{i,t'}\right) - b\left(s_{i,t}\right)\right)$$

Simple baseline: $b = \frac{1}{N}\sum_{n=1}^{N} r(\tau_n)$

average reward is not the best baseline, but it's pretty good!

Baseline removal

- **EGLP Lemma.** Suppose that p_θ is a parameterized probability distribution

$$E_{x \sim p_\theta(x)}[\nabla_\theta \log p_\theta(x)] = 0$$

- Therefore

$$E_{a \sim \pi_\theta(.|s)}[\nabla_\theta \log \pi_\theta(a|s)] = 0$$

$$\Longrightarrow E_{a \sim \pi_\theta(.|s)}[\nabla_\theta \log \pi_\theta(a|s)\, b(s)] = 0$$

The gradient estimates **without introducing bias**.

$$E\left[\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}) - b(s_{i,t})\right)\right] = E\left[\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'})\right]$$

Baseline removal

State-dependent expected return:

$$b(s_t) = E\left[r_t + r_{t+1} + \cdots + r_T | \pi\right] = V^{\pi}(s_t)$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta\left(a_{i,t}|s_{i,t}\right)\left(\sum_{t'=t}^{T} r\left(s_{i,t'}, a_{i,t'}\right) - V^{\pi}\left(s_{i,t}\right)\right)$$

Increase logprob of action proportionally to how much its returns are better than the expected return under the current policy

Approximate $V^{\pi}(s_t)$

- Approximate $V^{\pi}(s_t)$ with a model $V_{\phi}(s_t)$

- It is updated concurrently with the policy
  - Value network always approximates the value function of the most recent policy

$$\phi_k = \underset{\phi}{\operatorname{argmin}} E_{a_t \sim \pi_k, s_t} \left[\left(V_{\phi}(s_t) - y_t \right)^2 \right]$$

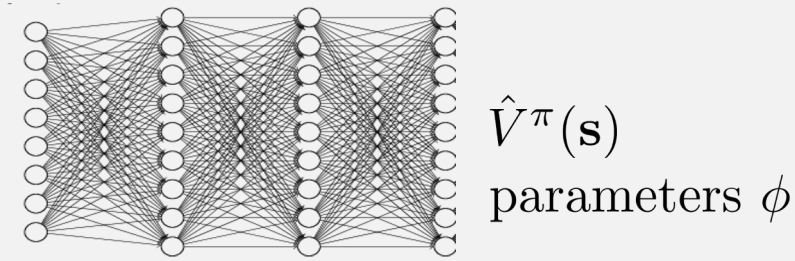
Monte Carlo Estimation of V^{π}

Goal: fit V^{π}

ideal target: $y_{i,t} = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_{i,t}\right]$

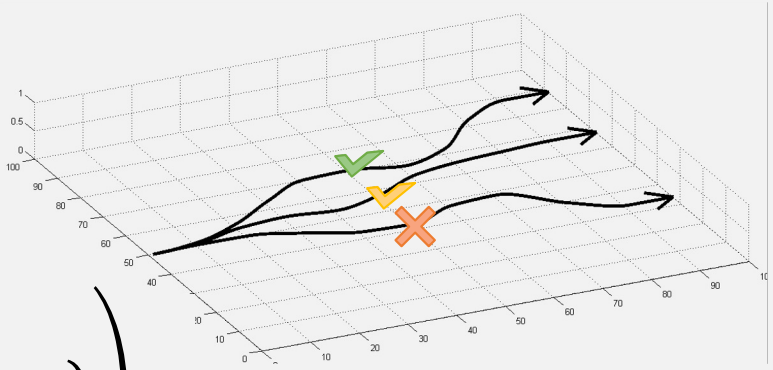
Monte Carlo target: $y_{i,t} = \sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})$

supervised regression: $\mathcal{L}(\phi) = \frac{1}{2} \sum_{i} \left\| \hat{V}_{\phi}^{\pi}(\mathbf{s}_i) - y_i \right\|^2$



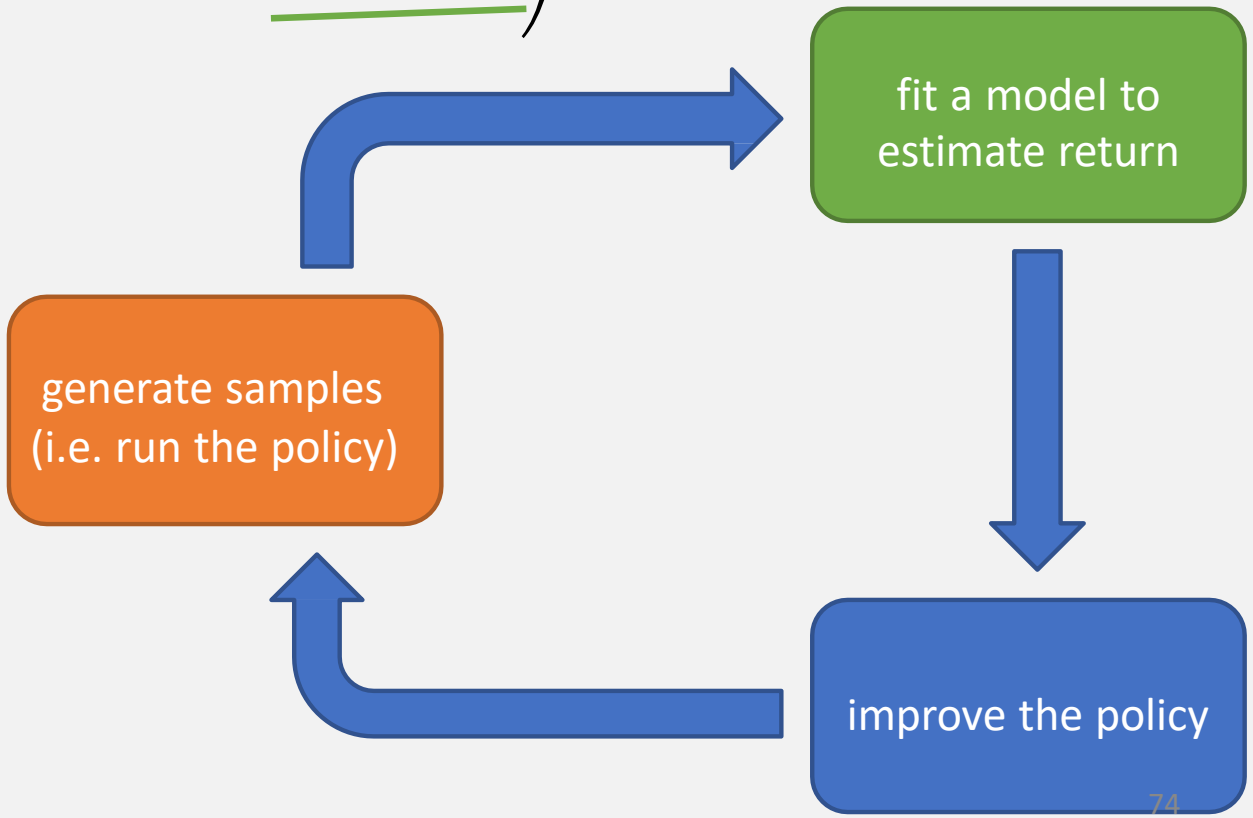
Slide adapted from Sergey Levine

Evaluating the policy gradient



$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}) - V^\pi(s_{i,t})\right)$$

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



Slide adapted from Sergey Levine

Vanilla Policy Gradient

Algorithm 1 "Vanilla" policy gradient algorithm

Initialize policy parameter θ, baseline b

for iteration=$1, 2, \ldots$ **do**

Collect a set of trajectories by executing the current policy

At each timestep in each trajectory, compute

the *return* $R_t = \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}$, and

the *advantage estimate* $\hat{A}_t = R_t - b(s_t)$.

Re-fit the baseline, by minimizing $\|b(s_t) - R_t\|^2$,

summed over all trajectories and timesteps.

Update the policy, using a policy gradient estimate $\hat{g}$,

which is a sum of terms $\nabla_\theta \log \pi(a_t \mid s_t, \theta) \hat{A}_t$

end for

~ [Williams, 1992]

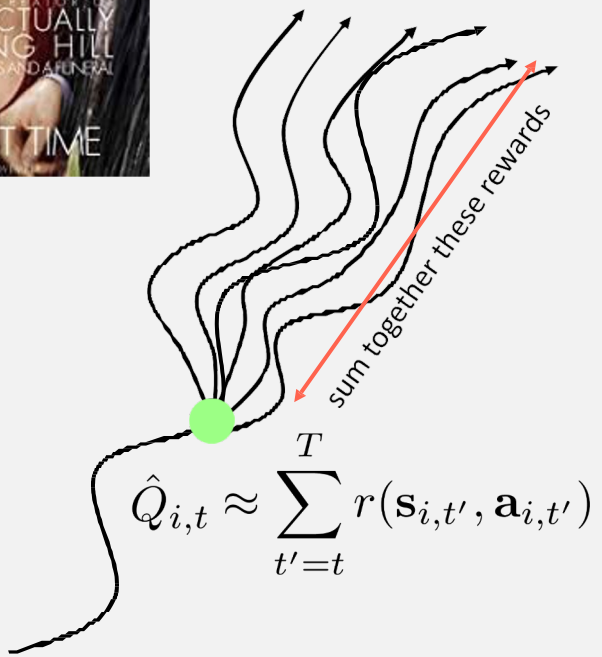
Improving the policy gradient

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \underbrace{\left(\sum_{t'=t}^{T} r(\mathbf{a}_{i,t'}, \mathbf{s}_{i,t'}) \right)}$$

Reward "to go"

$\hat{Q}_{i,t}$

$\hat{Q}_{i,t}$: estimate of expected reward if we take action $\mathbf{a}_{i,t}$ in state $\mathbf{s}_{i,t}$



Slide adapted from Sergey Levine

Advantage function

- relative **advantage** of an action in a state

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s).$$

Algorithm 1 Vanilla Policy Gradient Algorithm

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
6: Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\big|_{\theta_k} \hat{A}_t.$$

7: Compute policy update, either using standard gradient ascent,

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k,$$

or via another gradient ascent algorithm like Adam.
8: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

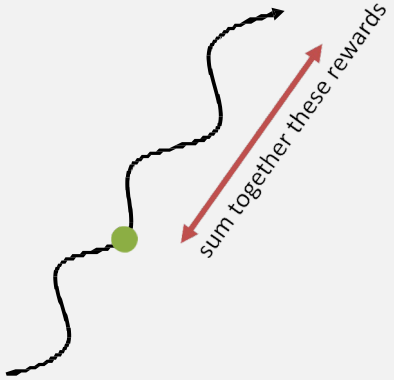
typically via some gradient descent algorithm.
9: **end for**

Monte Carlo Estimation of V^{π}

Goal: fit V^{π}

ideal target: $y_{i,t} = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_{i,t}\right]$

Monte Carlo target: $y_{i,t} = \sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})$



- How do you update your predictions about winning the game?
- What happens if you don't finish the game?
- Do you always wait till the end?

Slide adapted from Sergey Levine

Recap: Value functions

$V^{\pi}(s)$: How good for the agent to be in the state s when its policy is π

$$V^{\pi}(s_t) = \sum_{t'=t}^{T} \mathbb{E}[r(s_{t'}, a_{t'})|s_t, \pi]$$

Total reward starting from s and following policy π

$Q^{\pi}(s, a)$: How good for the agent to be in the state s and do action a when its policy is π

$$Q^{\pi}(s_t, a_t) = \sum_{t'=t}^{T} \mathbb{E}[r(s_{t'}, a_{t'})|s_t, a_t, \pi]$$

Total reward starting from s, taking a and then following policy π

$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi(.|s), s'\sim p(.|s,a)}[r(s, a) + \gamma V^{\pi}(s')]$$

$$Q^{\pi}(s, a) = \mathbb{E}_{s'\sim p(.|s,a)}[r(s, a) + \gamma Q^{\pi}(s', a')]$$

Bellman Equations

Bootstrap Estimation of V^{π}

Goal: fit V^{π}

ideal target: $y_{i,t} = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_{i,t}\right]$

Monte Carlo target: $y_{i,t} = \sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})$

Slide adapted from Sergey Levine

Bootstrap Estimation of V^{π}

Goal: fit V^{π}

ideal target: $y_{i,t} = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_{i,t}\right] \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \sum_{t'=t+1}^{T} E_{\pi_\theta}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_{i,t+1}]$

Monte Carlo target: $y_{i,t} = \sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})$

Slide adapted from Sergey Levine

Bootstrap Estimation of V^{π}

Goal: fit V^{π}

ideal target: $y_{i,t} = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_{i,t}\right] \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \sum_{t'=t+1}^{T} E_{\pi_\theta}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_{i,t+1}] \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + V^{\pi}(\mathbf{s}_{i,t+1})$

Monte Carlo target: $y_{i,t} = \sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})$

directly use previous fitted value function!

Slide adapted from Sergey Levine

Bootstrap Estimation of V^{π}

Goal: fit V^{π}

ideal target: $y_{i,t} = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_{i,t}\right] \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \sum_{t'=t+1}^{T} E_{\pi_\theta}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_{i,t+1}] \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \underbrace{V^{\pi}(\mathbf{s}_{i,t+1})}$

Monte Carlo target: $y_{i,t} = \sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})$

directly use previous fitted value function!

training data: $\left\{\left(\mathbf{s}_{i,t}, \underbrace{r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}_{\phi}^{\pi}(\mathbf{s}_{i,t+1})}_{y_{i,t}}\right)\right\}$

Slide adapted from Sergey Levine

Bootstrap Estimation of V^{π}

Goal: fit V^{π}

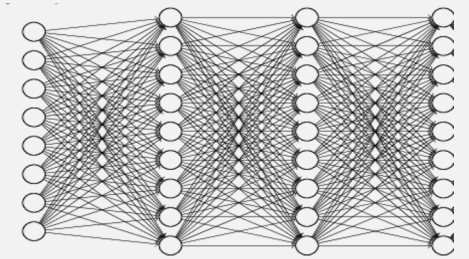
ideal target: $y_{i,t} = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_{i,t}\right] \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \sum_{t'=t+1}^{T} E_{\pi_\theta}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_{i,t+1}] \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + V^{\pi}(\mathbf{s}_{i,t+1})$

Monte Carlo target: $y_{i,t} = \sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})$

directly use previous fitted value function!

training data: $\left\{\left(\mathbf{s}_{i,t}, \underbrace{r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}_{\phi}^{\pi}(\mathbf{s}_{i,t+1})}_{y_{i,t}}\right)\right\}$

supervised regression: $\mathcal{L}(\phi) = \frac{1}{2}\sum_{i}\left\|\hat{V}_{\phi}^{\pi}(\mathbf{s}_i) - y_i\right\|^2$



sometimes referred to as a "bootstrapped" estimate

Slide adapted from Sergey Levine

Actor-Critic

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

online actor-critic algorithm:

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
2. update $\hat{V}_\phi^\pi$ using target $r + \gamma \hat{V}_\phi^\pi(\mathbf{s}')$
3. evaluate $\hat{A}^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}') - \hat{V}_\phi^\pi(\mathbf{s})$
4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \hat{A}^\pi(\mathbf{s}, \mathbf{a})$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$