Q learning

CS 224R

Reminders

Today:	Project survey due
Wednesday:	Homework 1 due, Homework 2 out

The Plan

Advanced policy gradients recap

Actor-critic & case studies

Policy iteration and value iteration

Q learning

Key learning goals:

- Understand the difference between policy & value iteration
- Intuition of Q learning

The Plan

Advanced policy gradients recap

Actor-critic & case studies

Policy iteration and value iteration

Q learning

Policy gradients

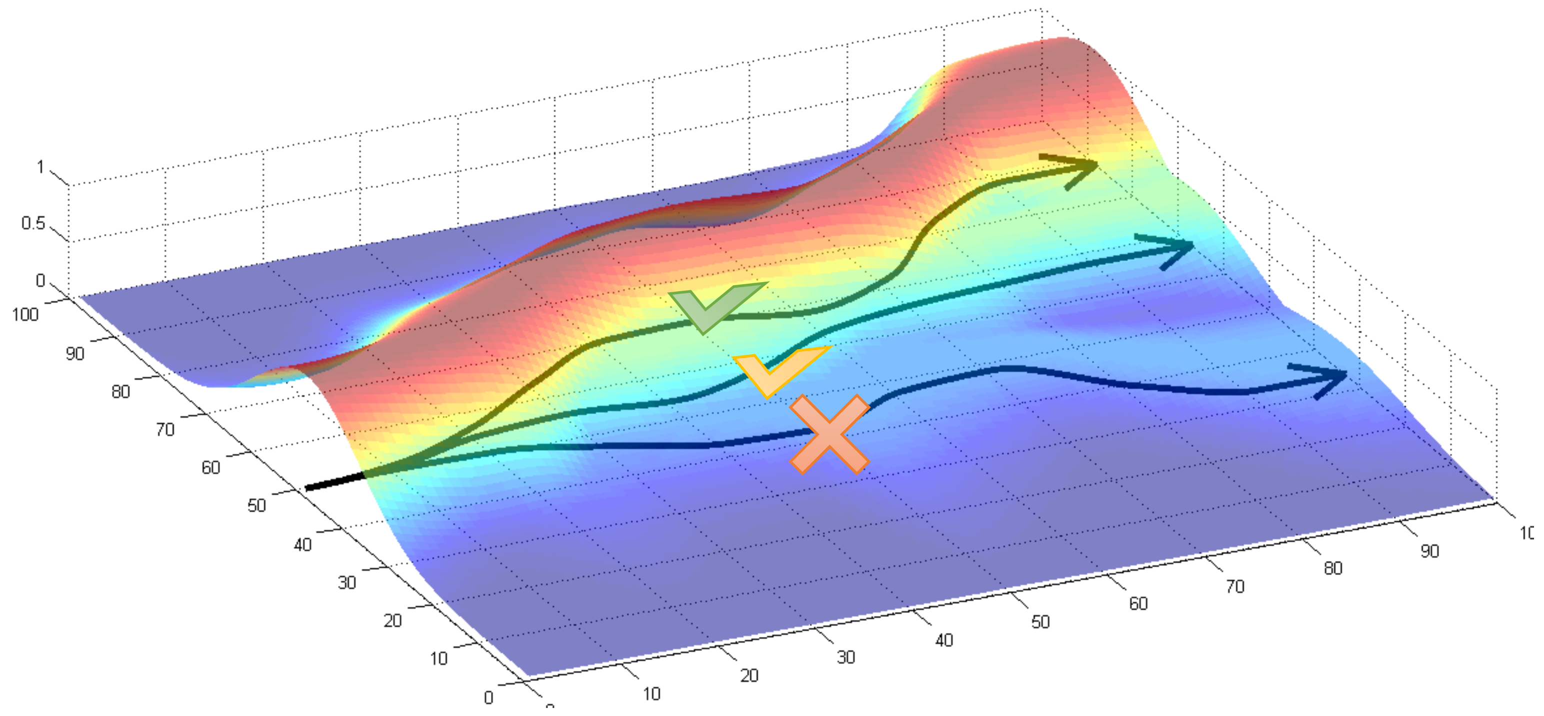
$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)$$

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

good stuff is made more likely

bad stuff is made less likely

simply formalizes the notion of "trial and error"!

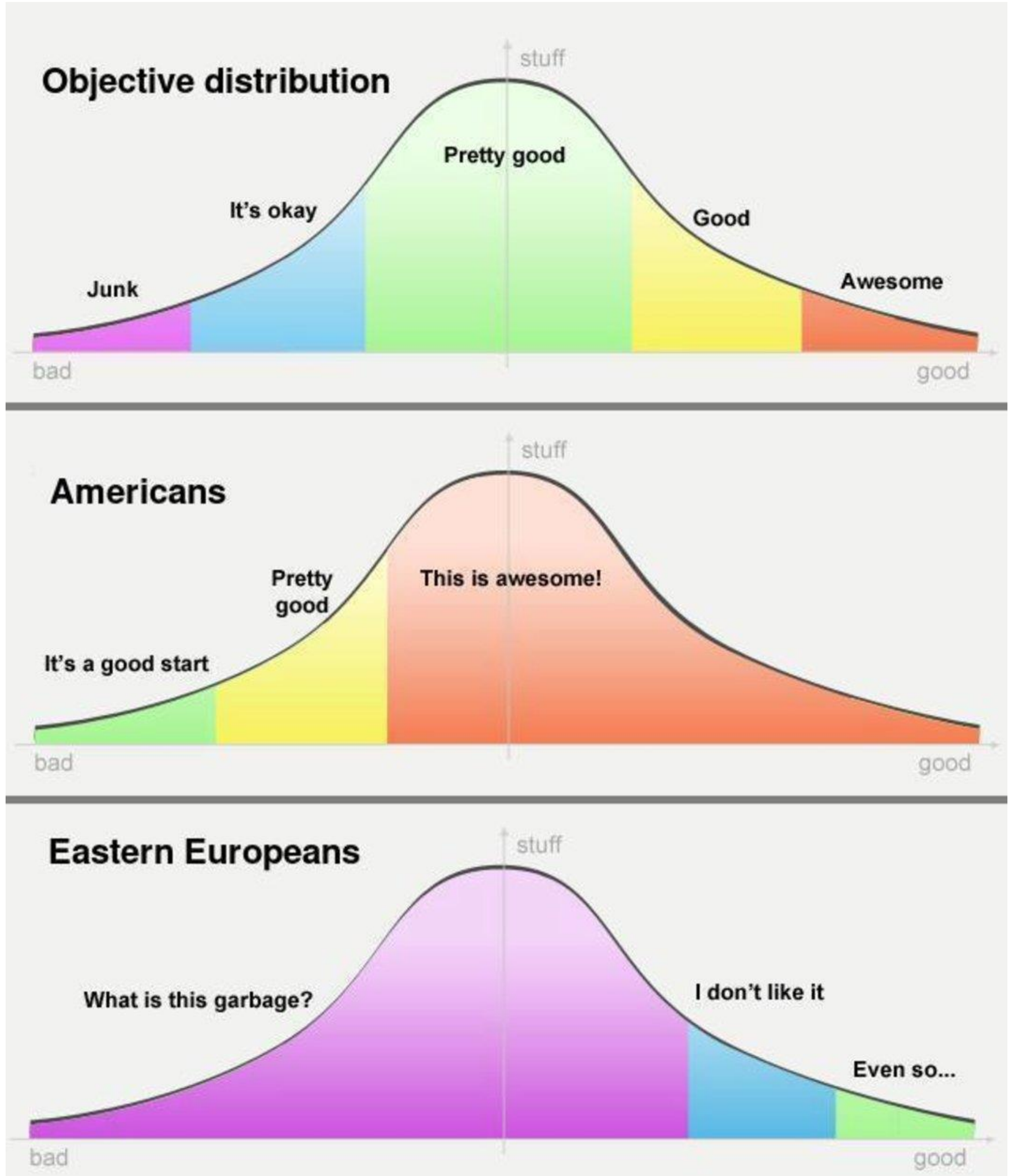


REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
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. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Variance of the gradient estimator

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

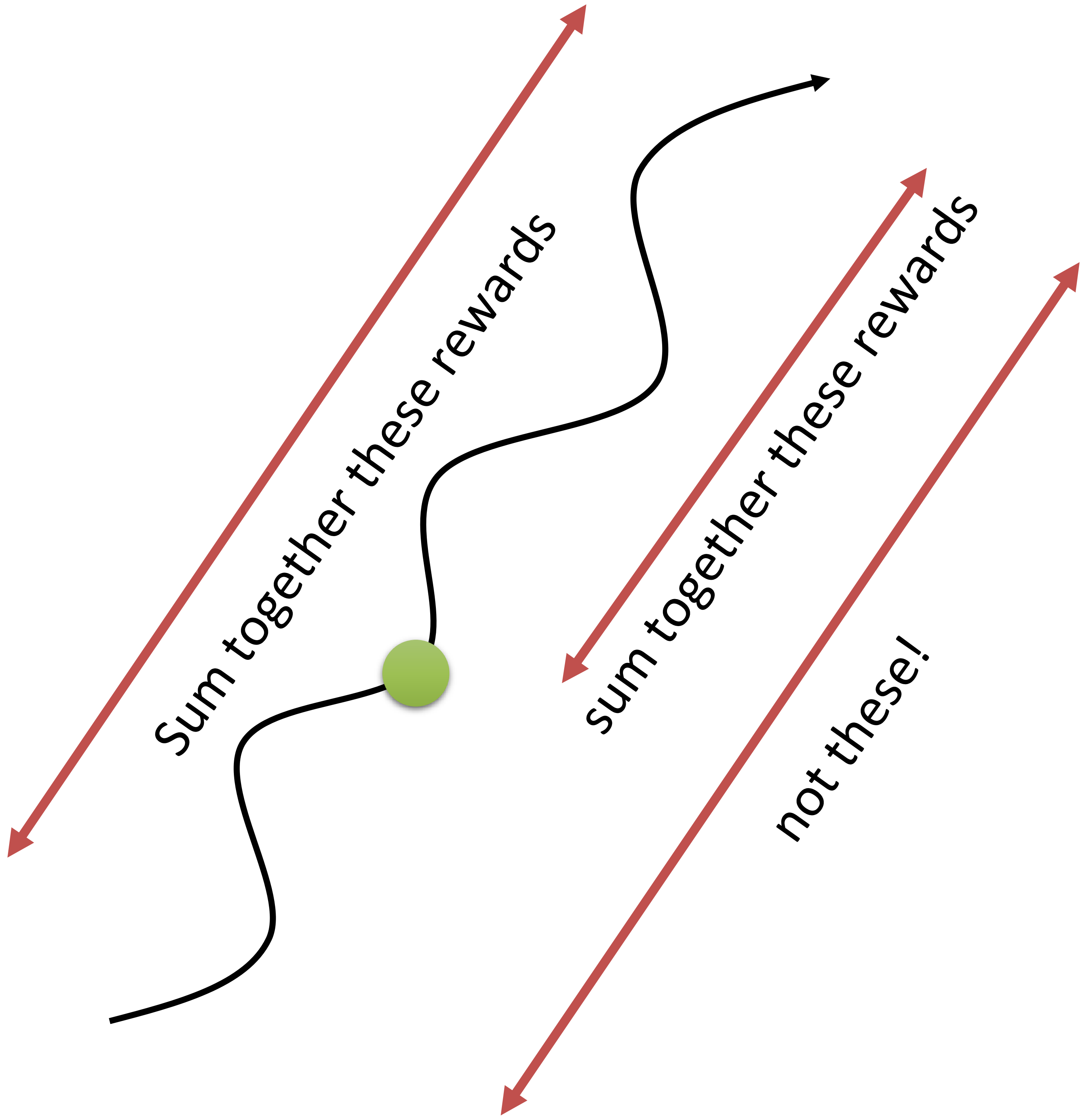


Small way to reduce variance

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

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

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



Improving the policy gradient

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

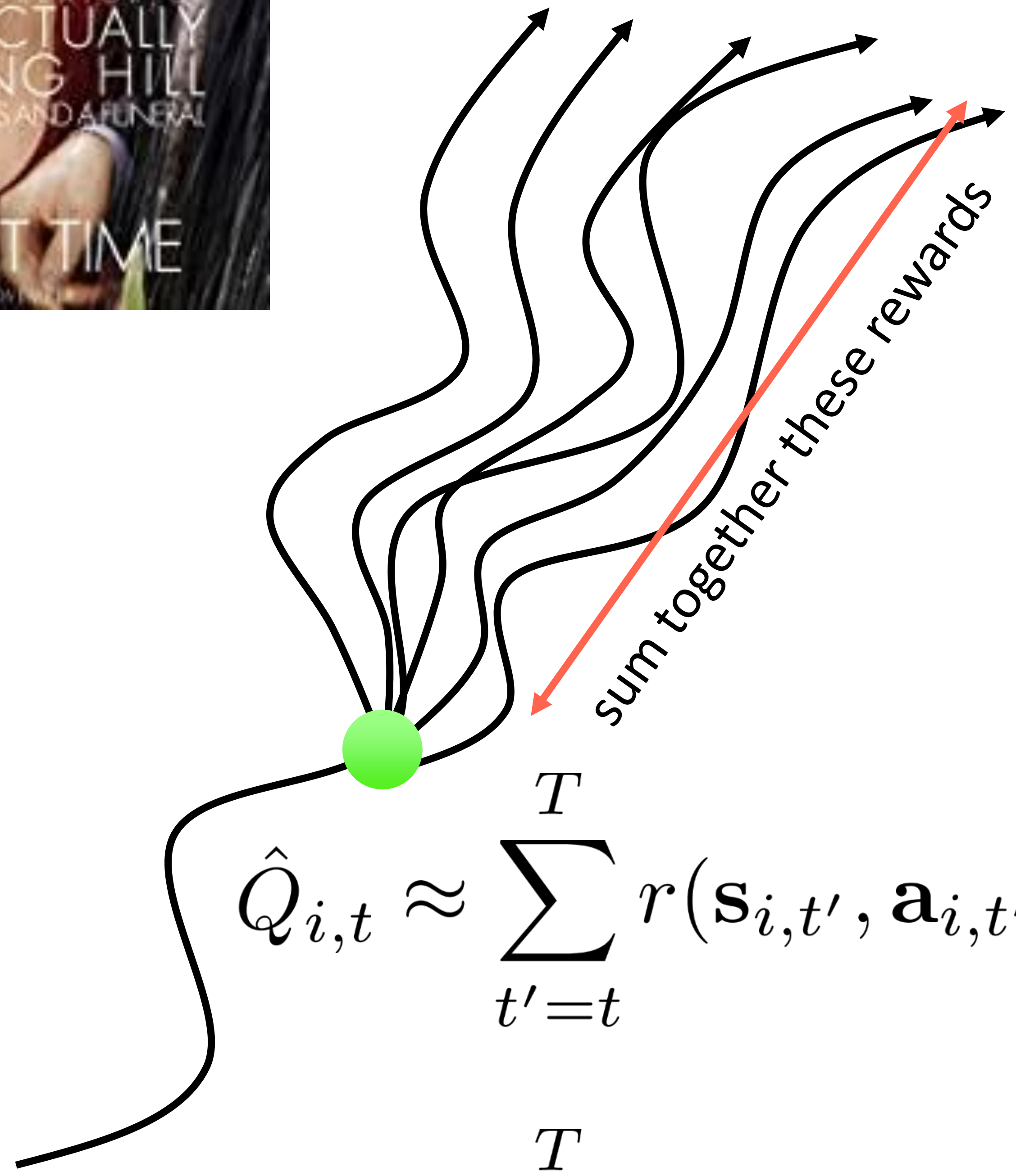
$$\hat{Q}_{i,t}$$

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

can we get a better estimate?

$Q(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t, \mathbf{a}_t\right]$: true *expected* reward-to-go

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



$$\hat{Q}_{i,t} \approx \sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})$$

$$\hat{Q}_{i,t} \approx \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t, \mathbf{a}_t\right]$$

Slide adapted from Sergey Levine

State & state-action value functions

$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t\right]$: total reward from taking $\mathbf{a}_t$ in $\mathbf{s}_t$

$V^\pi(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)}[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)]$: total reward from $\mathbf{s}_t$

$A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)$: how much better $\mathbf{a}_t$ is

Slide adapted from Sergey Levine

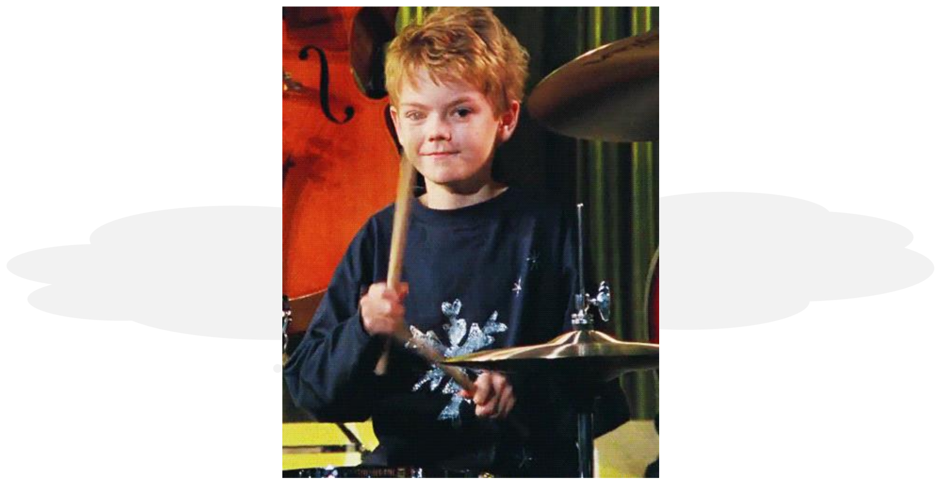
Value-Based RL

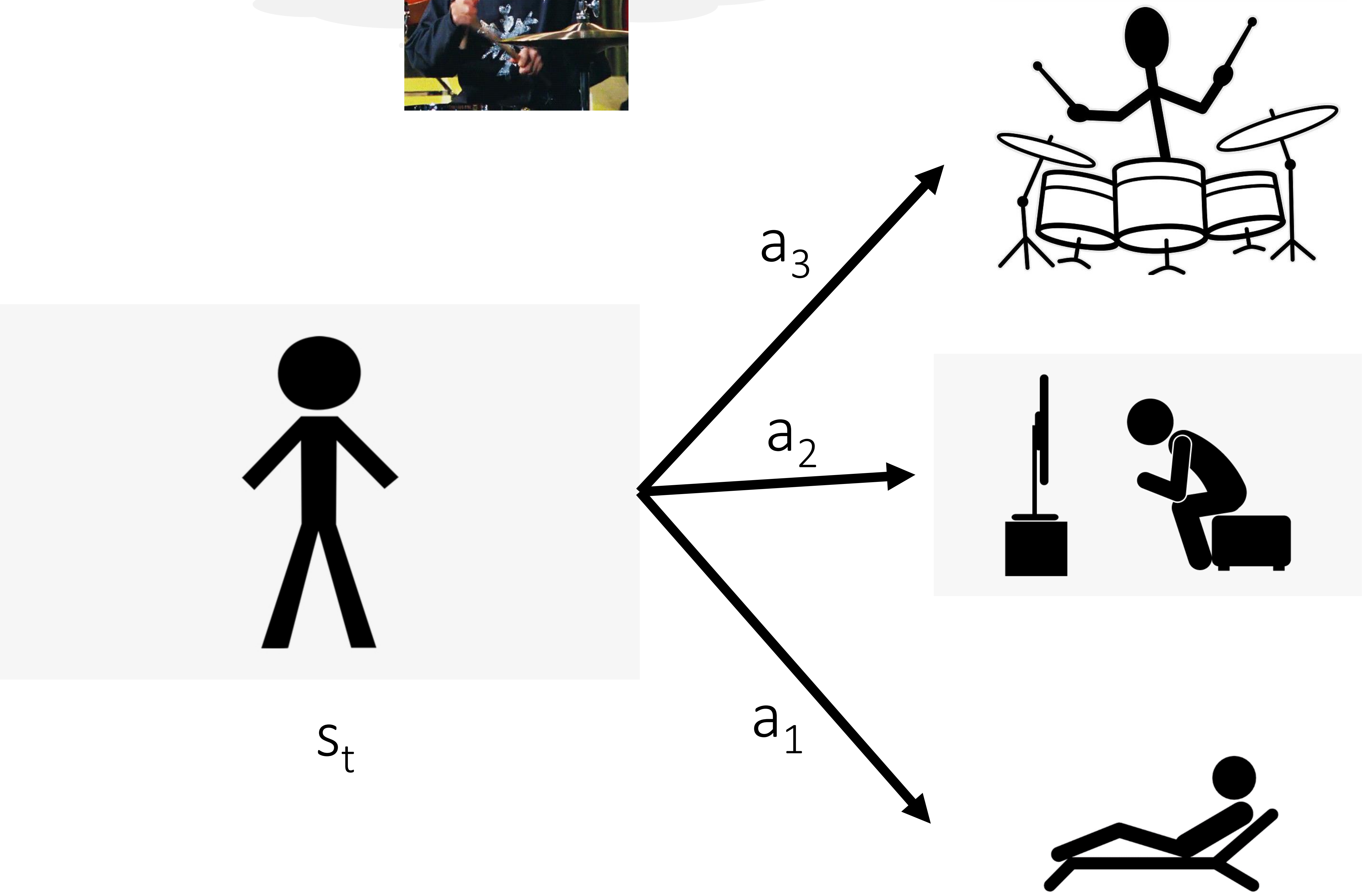
Value function: $V^{\pi}(\mathbf{s}_t) = ?$

Q function: $Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = ?$

Advantage function: $A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = ?$

Reward = 1 if I can play it in a month, 0 otherwise





Current $\pi(\mathbf{a}_1|\mathbf{s}) = 1$

How can we use all of this to fit a better estimator?

Goal: fit V^π

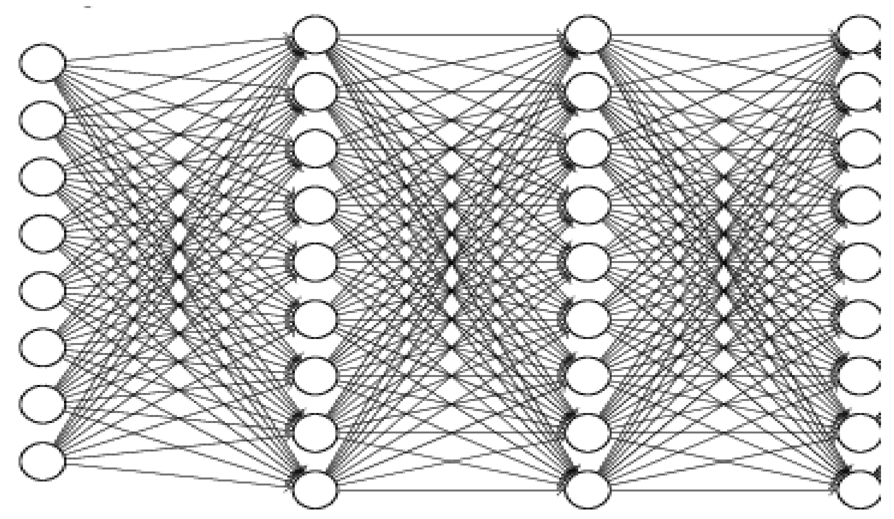
ideal target: $y_{i,t} = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_{i,t}\right] \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + V^\pi(\mathbf{s}_{i,t+1}) \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \underbrace{\hat{V}^\pi_\phi(\mathbf{s}_{i,t+1})}_{\text{directly use previous fitted value function!}}$

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

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

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

$\mathbf{s}$



$\hat{V}^\pi(\mathbf{s})$

parameters ϕ

sometimes referred to as a "bootstrapped" estimate

Problem with importance sampling in (policy gradient)

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim \pi_\theta(\tau)} \left[\left(\frac{\prod_{t=1}^T \pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\prod_{t=1}^T \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \right) \left(\sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \right) \left(\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

Importance sampling

$$E_{x \sim p(x)}[f(x)] = \int p(x) f(x) dx$$

$$\int p(x) \frac{q(x)}{q(x)} f(x) dx = \int q(x) \frac{p(x)}{q(x)} f(x) dx$$

$$E_{x \sim q(x)}\left[\frac{p(x)}{q(x)} f(x)\right]$$

$$\theta' \leftarrow \arg\max(\theta' - \theta) \nabla_\theta J(\theta) \quad s.t. D_{KL}(\pi_{\theta'}, \pi_\theta) \leq \epsilon$$

Proximal policy optimization (PPO)

Apply all the tricks:

- Use advantage function to reduce the variance
- Use importance sampling to take multiple gradient steps
- Constrain the optimization objective in the policy space

Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI
{joschu, filip, prafulla, alec, oleg}@openai.com

Let $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$, so $r(\theta_{\text{old}}) = 1$. TRPO maximizes a "surrogate" objective

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]. \tag{6}$$

The superscript CPI refers to conservative policy iteration [KL02], where this objective was proposed. Without a constraint, maximization of L^{CPI} would lead to an excessively large policy update; hence, we now consider how to modify the objective, to penalize changes to the policy that move $r_t(\theta)$ away from 1.

The main objective we propose is the following:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \tag{7}$$

The Plan

Advanced policy gradients recap

Actor-critic & case studies

Policy iteration and value iteration

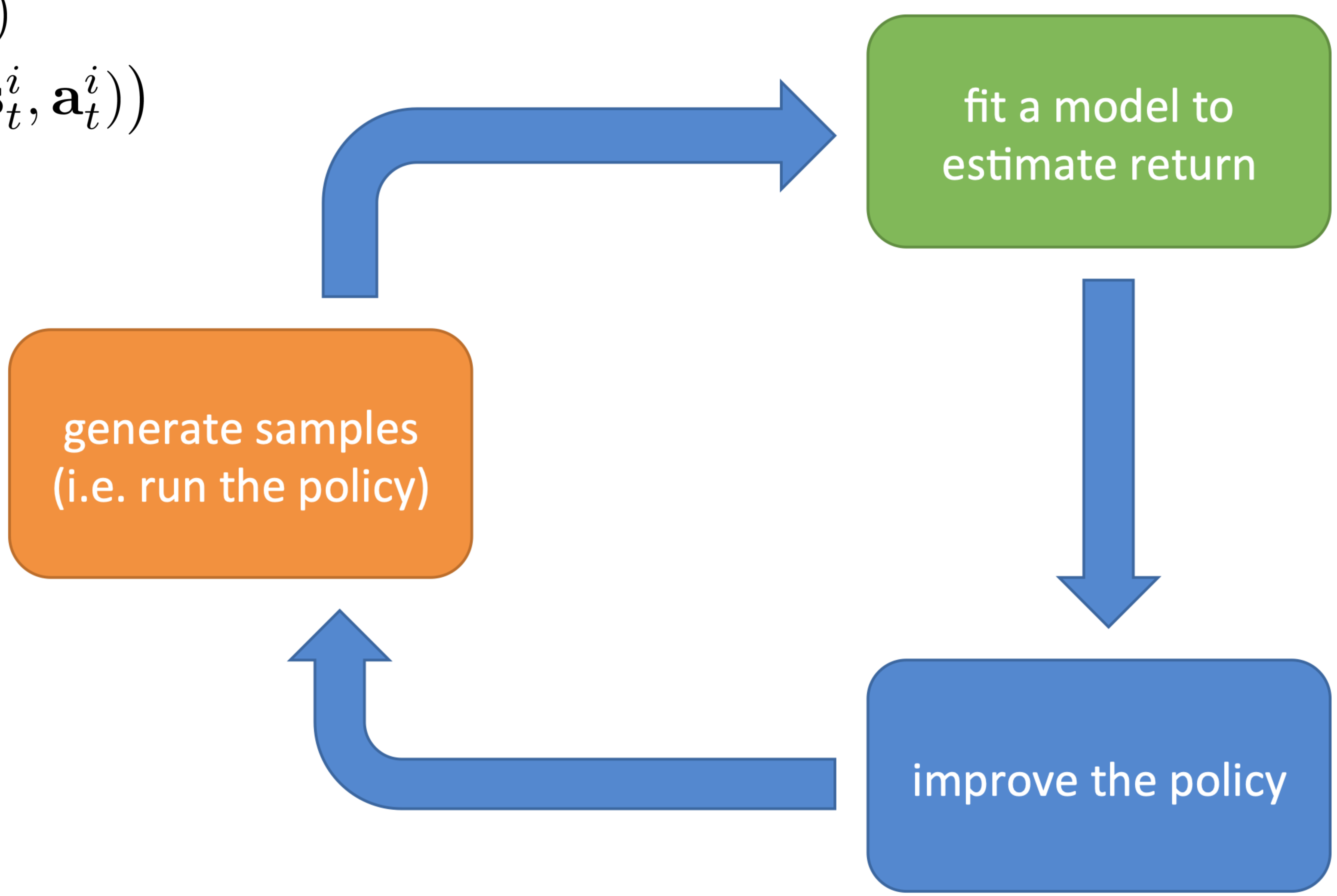
Q learning

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:

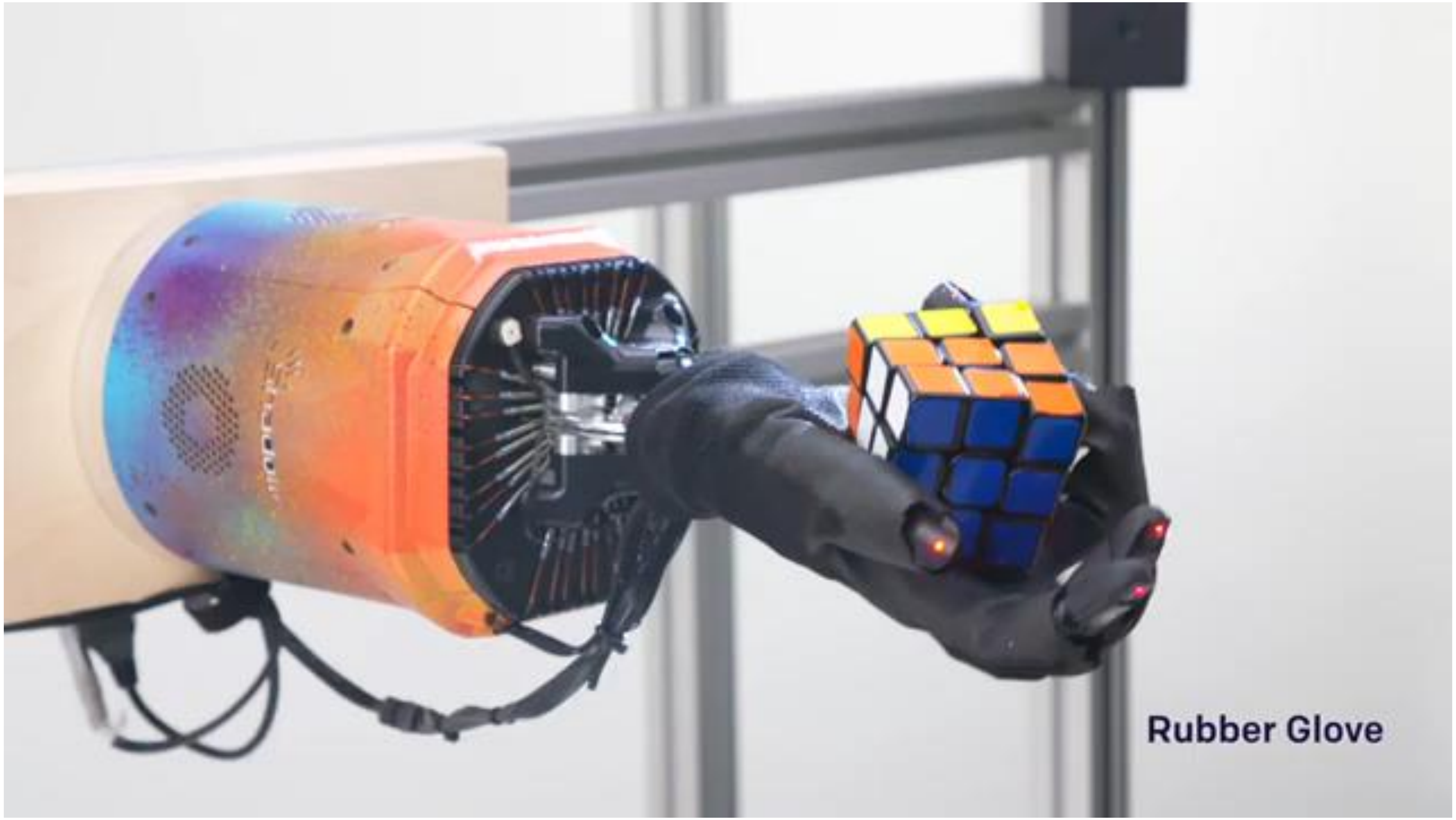
1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
2. update $\hat{V}^\pi_\phi$ using target $r + \gamma \hat{V}^\pi_\phi(\mathbf{s}')$
3. evaluate $\hat{A}^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \hat{V}^\pi_\phi(\mathbf{s}')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



Can we make it more off-policy friendly?

Slide adapted from Sergey Levine

Case study: PPO applied to robotics



Case study: PPO applied to robotics

Solving the Rubik's Cube with a robot hand is still not easy. Our method currently solves the Rubik's

We train neural networks to solve the Rubik's Cube in simulation using reinforcement learning and Kociemba's algorithm for picking the solution steps.[A] Domain randomization enables networks trained solely in simulation to transfer to a real robot.

into the hand and continue solving.

Simulator physics. We randomize simulator physics parameters such as geometry, friction, gravity, etc. See Section B.1 for details of their ADR parameterization.

Custom physics. We model additional physical robot effects that are not modelled by the simulator, for example, action latency or motor backlash. See [77, Appendix C.2] for implementation details of these models. We randomize the parameters in these models in a similar way to simulator physics randomizations.

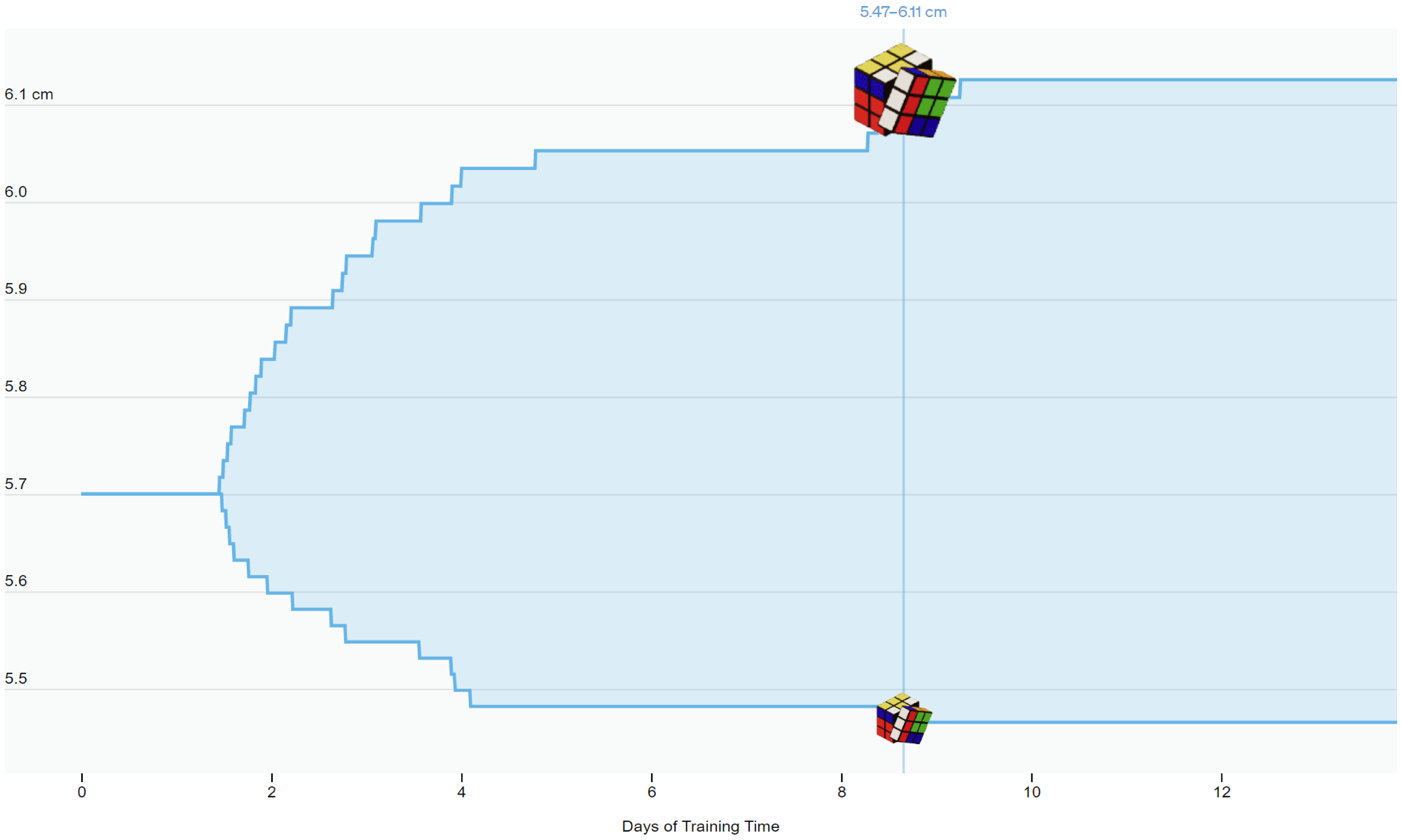
Adversarial. We use an adversarial approach similar to [82, 83] to capture any remaining unmodeled physical effects in the target domain. However, we use random networks instead of a trained adversary. See Section B.3 for details on implementation and ADR parameterization.

Observation. We add Gaussian noise to policy observations to better approximate observation conditions in reality. We apply both correlated noise, which is sampled once at the start of an episode and uncorrelated noise, which is sampled at each time step. We randomize the parameters of the added noise. See Section B.4 for details of their ADR parameterization.

Vision. We randomize several aspects in ORRB [16] to control the rendered scene, including lighting conditions, camera positions and angles, materials and appearances of all the objects, the texture of the background, and the post-processing effects on the rendered images. See Section B.5 for details.

Case study: PPO applied to robotics

ADR applied to the size of the Rubik's Cube



Case study: PPO applied to robotics

Table 6: Performance of different policies on the Rubik's cube for a fixed fair scramble goal sequence. We evaluate each policy on the real robot (N=10 trials) and report the mean ± standard error and median number of successes (meaning the total number of successful rotations and flips). We also report two success rates for applying half of a fair scramble ("half") and the other one for fully applying it ("full"). For ADR policies, we report the entropy in nats per dimension (npd). For "Manual DR", we obtain an upper bound on its ADR entropy by running ADR with the policy fixed and report the entropy once the distribution stops changing (marked with an "*").

Policy	**Sensing**		**ADR Entropy**	**Successes (Real)**		**Success Rate**	
	Pose	**Face Angles**		**Mean**	**Median**	**Half**	**Full**
Manual DR	Vision	Giiker	-0.569^{*} npd	1.8 ± 0.4	2.0	0 %	0 %
ADR	Vision	Giiker	-0.084 npd	3.8 ± 1.0	3.0	0 %	0 %
ADR (XL)	Vision	Giiker	0.467 npd	17.8 ± 4.2	12.5	30 %	10 %
ADR (XXL)	Vision	Giiker	**0.479 npd**	**26.8 ± 4.9**	**22.0**	**60 %**	**20 %**
ADR (XXL)	Vision	Vision	**0.479 npd**	12.8 ± 3.4	10.5	20 %	0 %

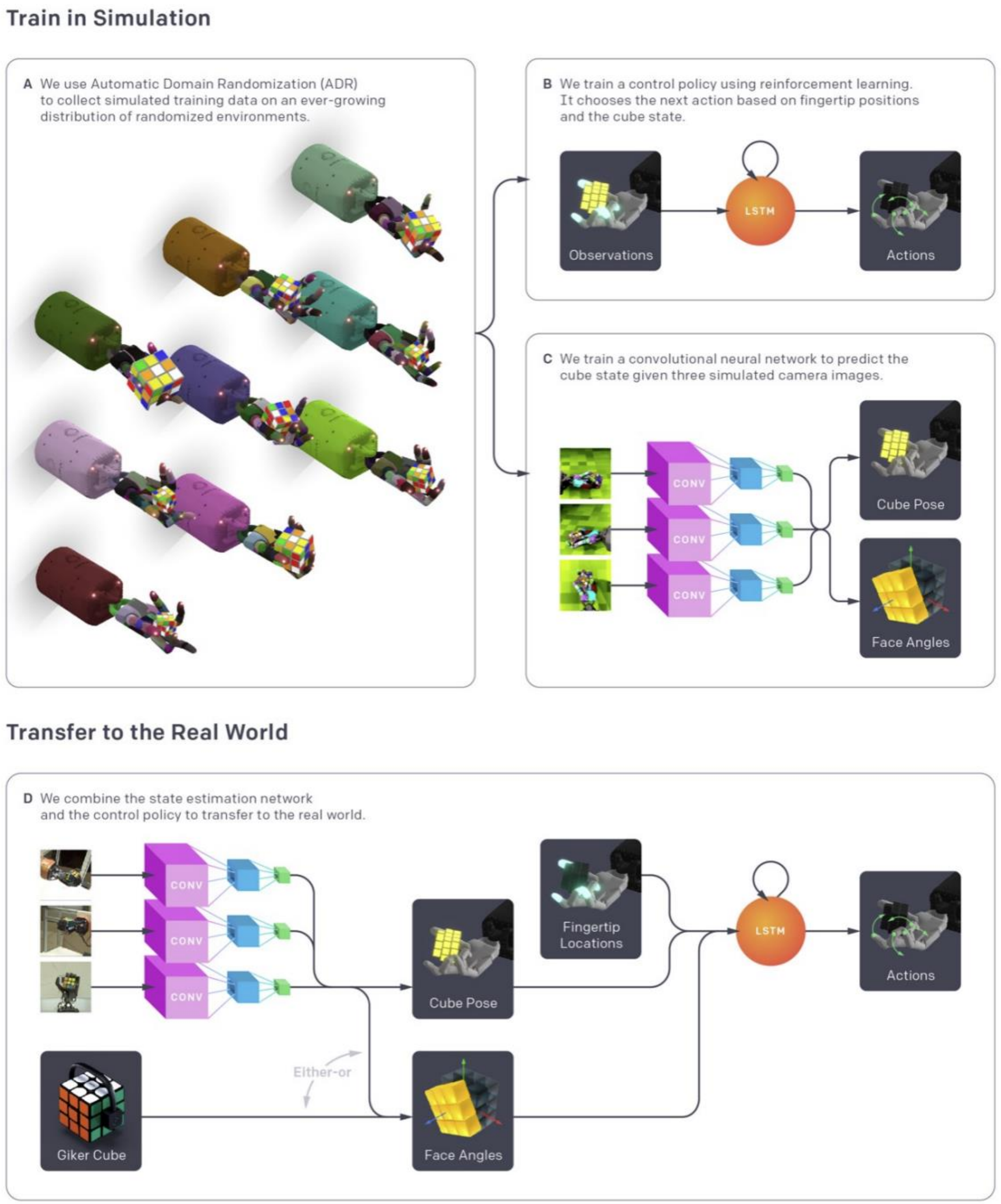
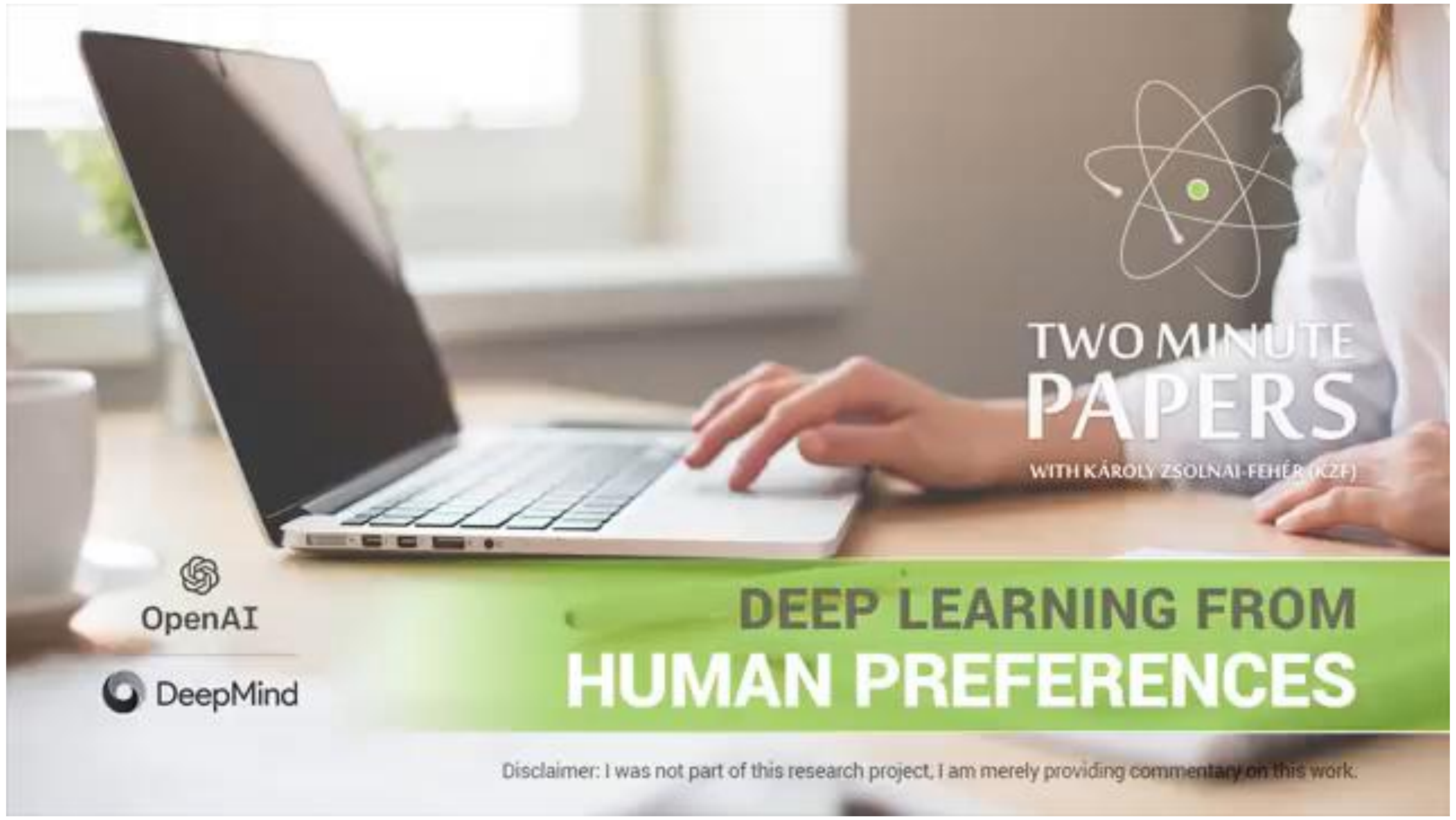
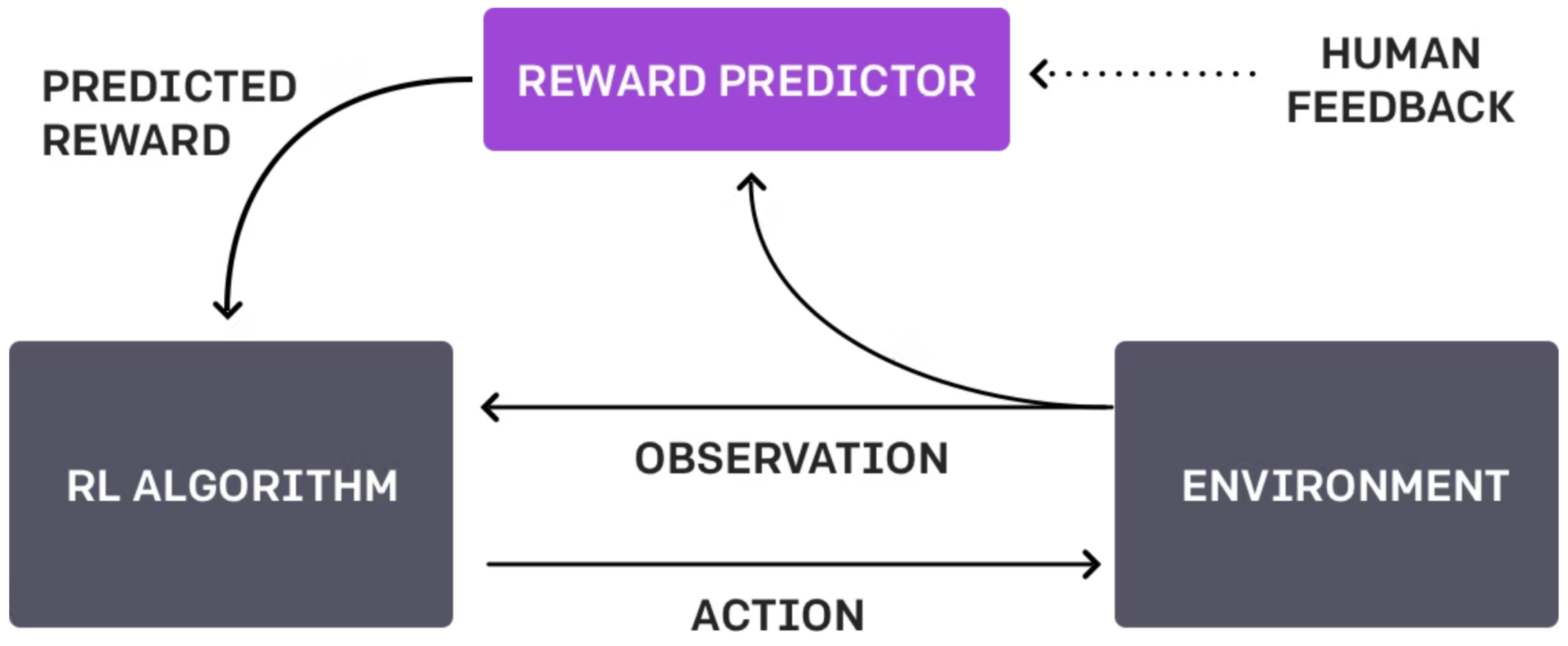


Figure 2: System Overview. (a) We use automatic domain randomization (ADR) to generate a growing distribution of simulations with randomized parameters and appearances. We use this data for both the control policy and vision-based state estimator. (b) The control policy receives observed robot states and rewards from the randomized simulations and learns to solve them using a recurrent neural network and reinforcement learning. (c) The vision-based state estimator uses rendered scenes collected from the randomized simulations and learns to predict the pose as well as face angles of the Rubik's cube using a convolutional neural network (CNN), trained separately from the control policy. (d) To transfer to the real world, we predict the Rubik's cube's pose from 3 real camera feeds with the CNN and measure the robot fingertip locations using a 3D motion capture system. The face angles that describe the internal rotational state of the Rubik's cube are provided by either the same vision state estimator *or* the Giiker cube, a custom cube with embedded sensors and feed it into the policy network.

Case study: PPO applied to LLMs (**speculations**)

RL from human feedback (RLHF)





Reward model

- Imagine a reward function: $R(s; p) \in \mathbb{R}$ for any output s to prompt p
- The reward is higher when humans prefer the output

SAN FRANCISCO, California (CNN) -- A magnitude 4.2 earthquake shook the San Francisco ... overturn unstable objects.	**An earthquake hit San Francisco. There was minor property damage, but no injuries.**	**The Bay Area has good weather but is prone to earthquakes and wildfires.**
	s_1	s_2
	$R(s_1; p) = 0.8$	$R(s_2; p) = 1.2$

Regularizing pre-trained model

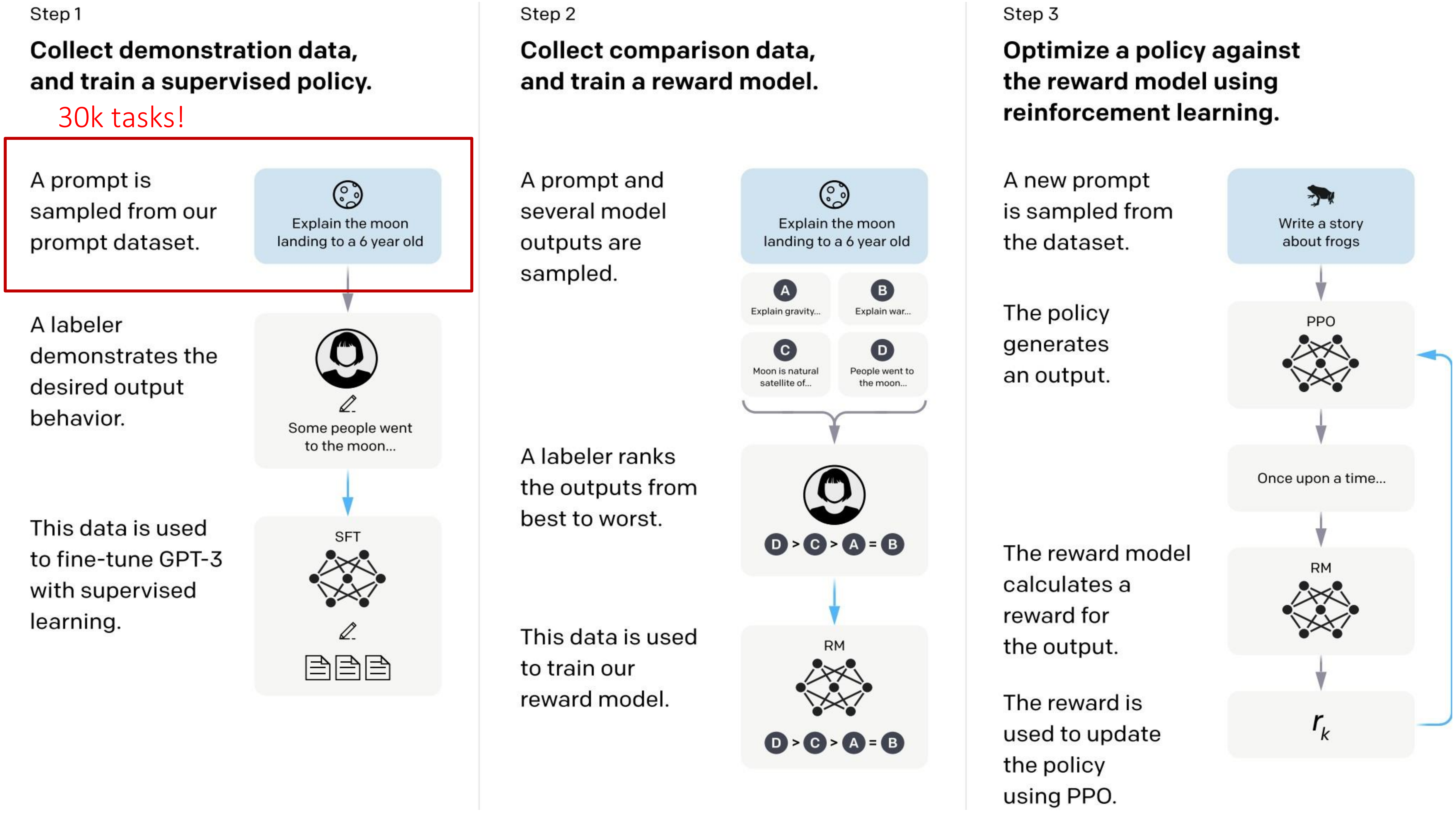
- **Challenge:** how do we ensure that $R(s;p)$ prefer natural language generations?
- Since $R(s;p)$ is trained on natural language inputs, it might fail to assign low scores to unnatural s.
- **Solution:** add regularization term to $R(s;p)$ that penalizes outputs that deviate from natural language.

$$\hat{R}(s;p) \coloneqq R(s;p) - \beta \log\left(\frac{p^{RL}(s)}{p^{PT}(s)}\right)$$

pay a price when $p^{RL}(s) < p^{PT}(s)$

- This is a penalty which prevents us from diverging too far from the pretrained model.

RLHF + PPO (speculations)



Case study: PPO applied to LLMs (**speculations**)

- Is HF more important than RL?
- What would you try next?
- What do you think the challenges are?

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
```

The Plan

Advanced policy gradients recap

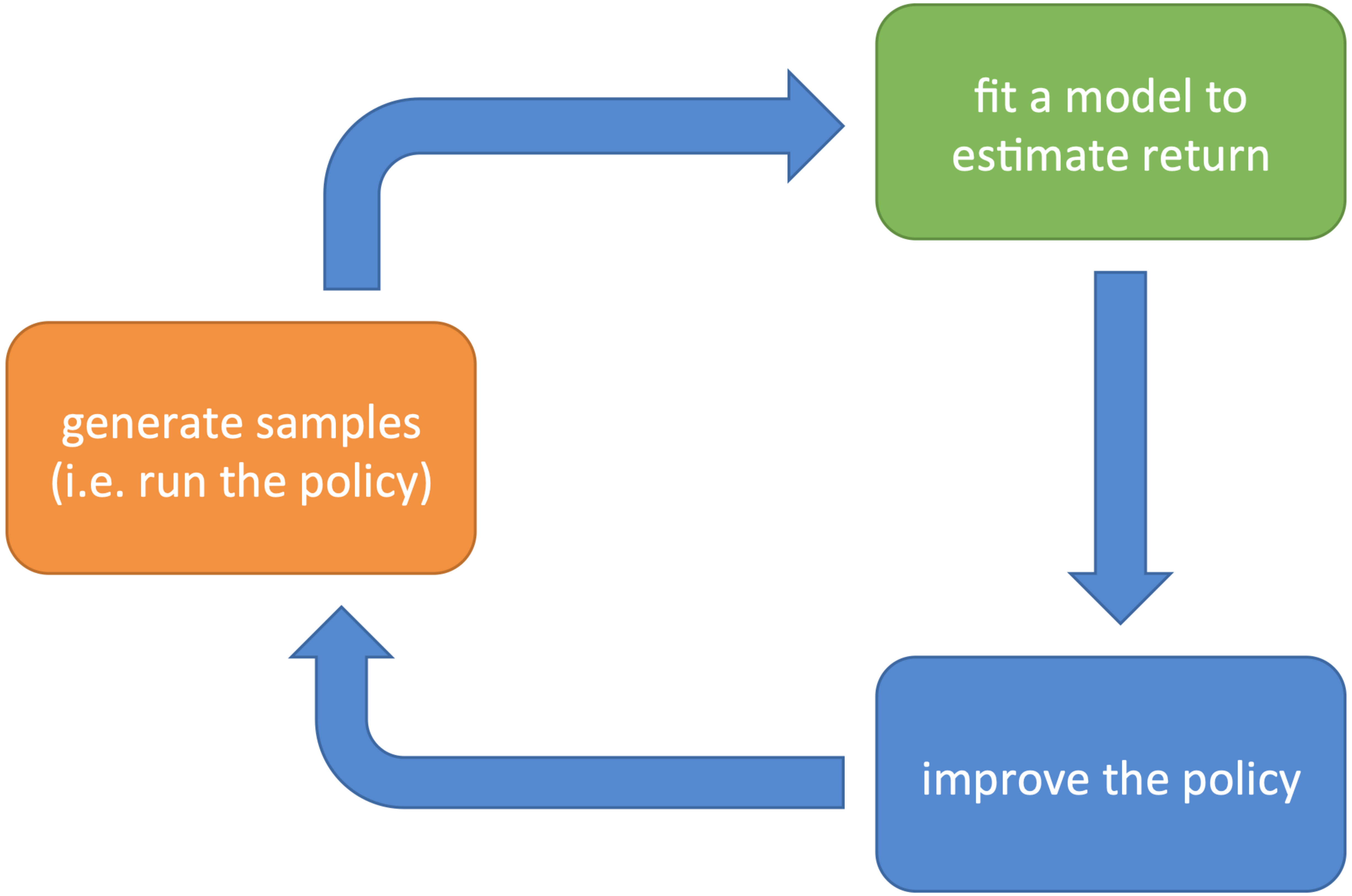
Actor-critic & case studies

Policy iteration and value iteration

Q learning

This was just the prediction part…

The anatomy of a reinforcement learning algorithm



compute $\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ (MC policy gradient)
fit $Q_\phi(\mathbf{s}, \mathbf{a})$ (actor-critic, Q-learning)
estimate $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ (model-based)

$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ (policy gradient)
$\pi(\mathbf{s}) = \arg\max Q_\phi(\mathbf{s}, \mathbf{a})$ (Q-learning)
optimize $\pi_\theta(\mathbf{a}|\mathbf{s})$ (model-based)

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

Improving the Policy

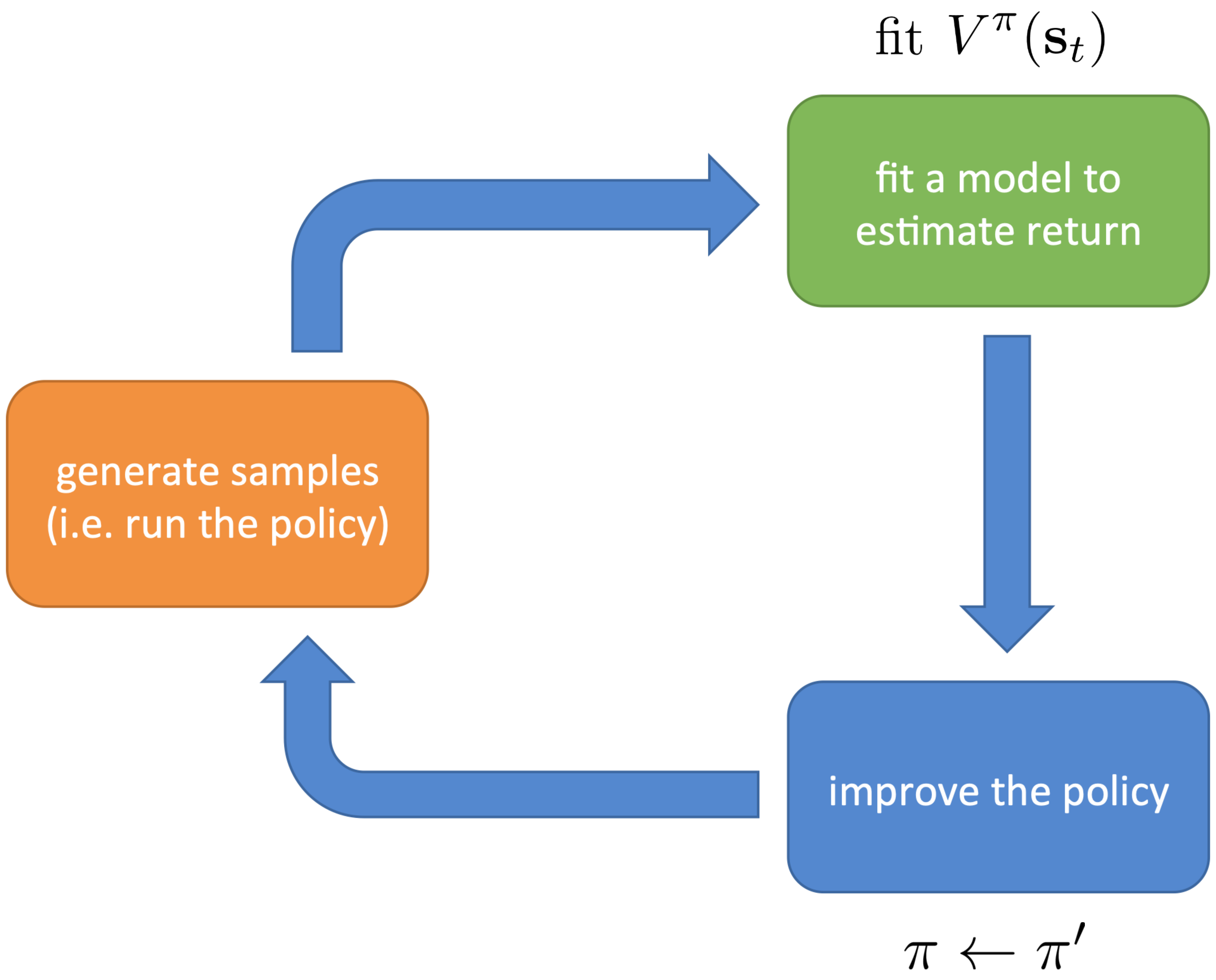
$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \text{ (policy gradient)}$$

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

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

$$\nabla_\theta J(\theta) \approx \quad \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \hat{A}^\pi(\mathbf{s}, \mathbf{a})$$

how good is an action compared to the policy?



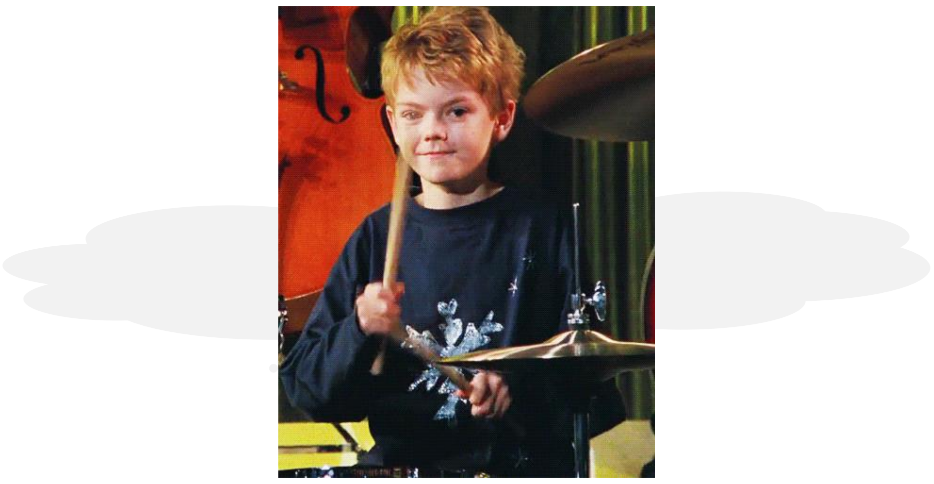
Value-Based RL

Value function: $V^{\pi}(\mathbf{s}_t) = ?$

Q function: $Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = ?$

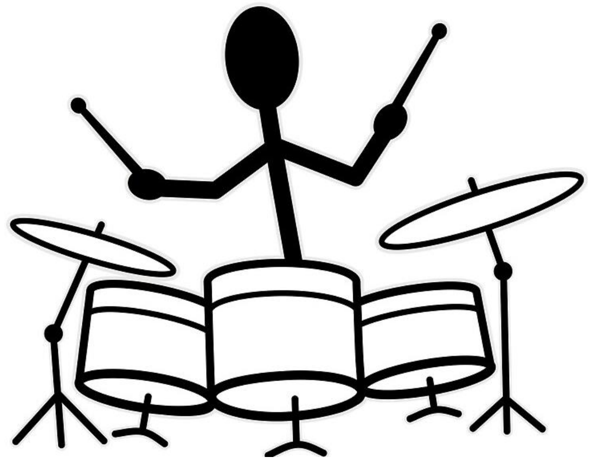
Advantage function: $A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = ?$

Reward = 1 if I can play it in a month, 0 otherwise

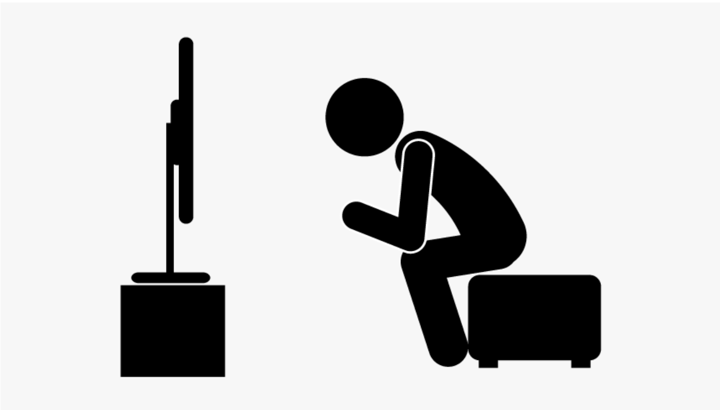


s_t

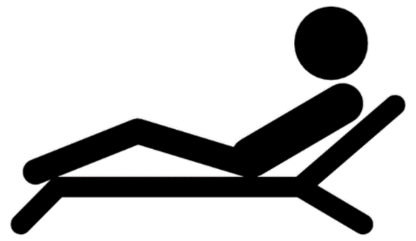
a_3



a_2



a_1



How can we improve the policy?

Current $\pi(\mathbf{a}_2|\mathbf{s}) = 1$

Improving the Policy

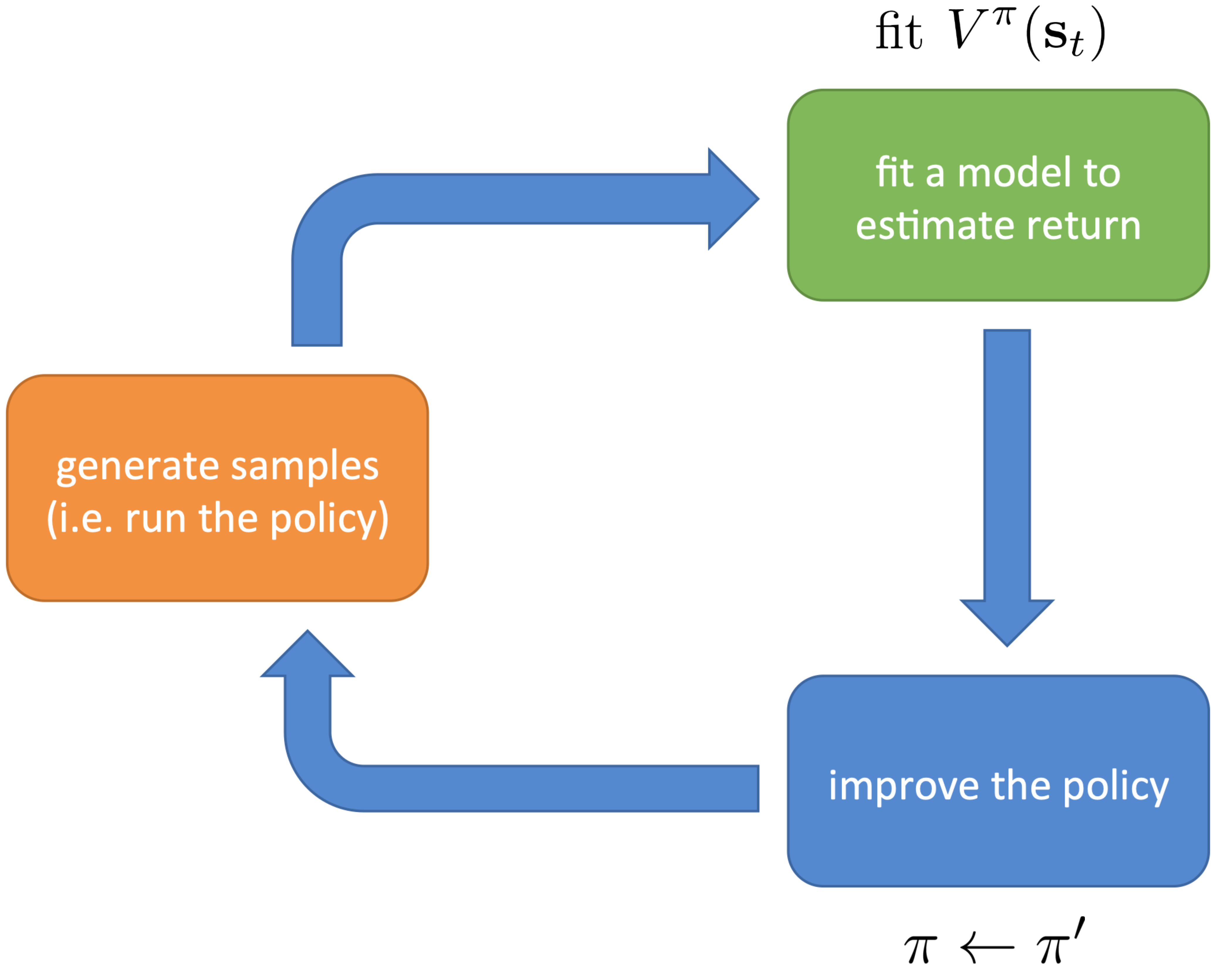
$A^\pi(\mathbf{s}_t, \mathbf{a}_t)$: how much better is $\mathbf{a}_t$ than the average action according to π

$\arg\max_{\mathbf{a}_t} A^\pi(\mathbf{s}_t, \mathbf{a}_t)$: best action from $\mathbf{s}_t$, if we then follow π

at *least* as good as any $\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$

regardless of what $\pi(\mathbf{a}_t|\mathbf{s}_t)$ is!

$$\pi'(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} A^\pi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 \text{ otherwise} \end{cases}$$



Slide adapted from Sergey Levine

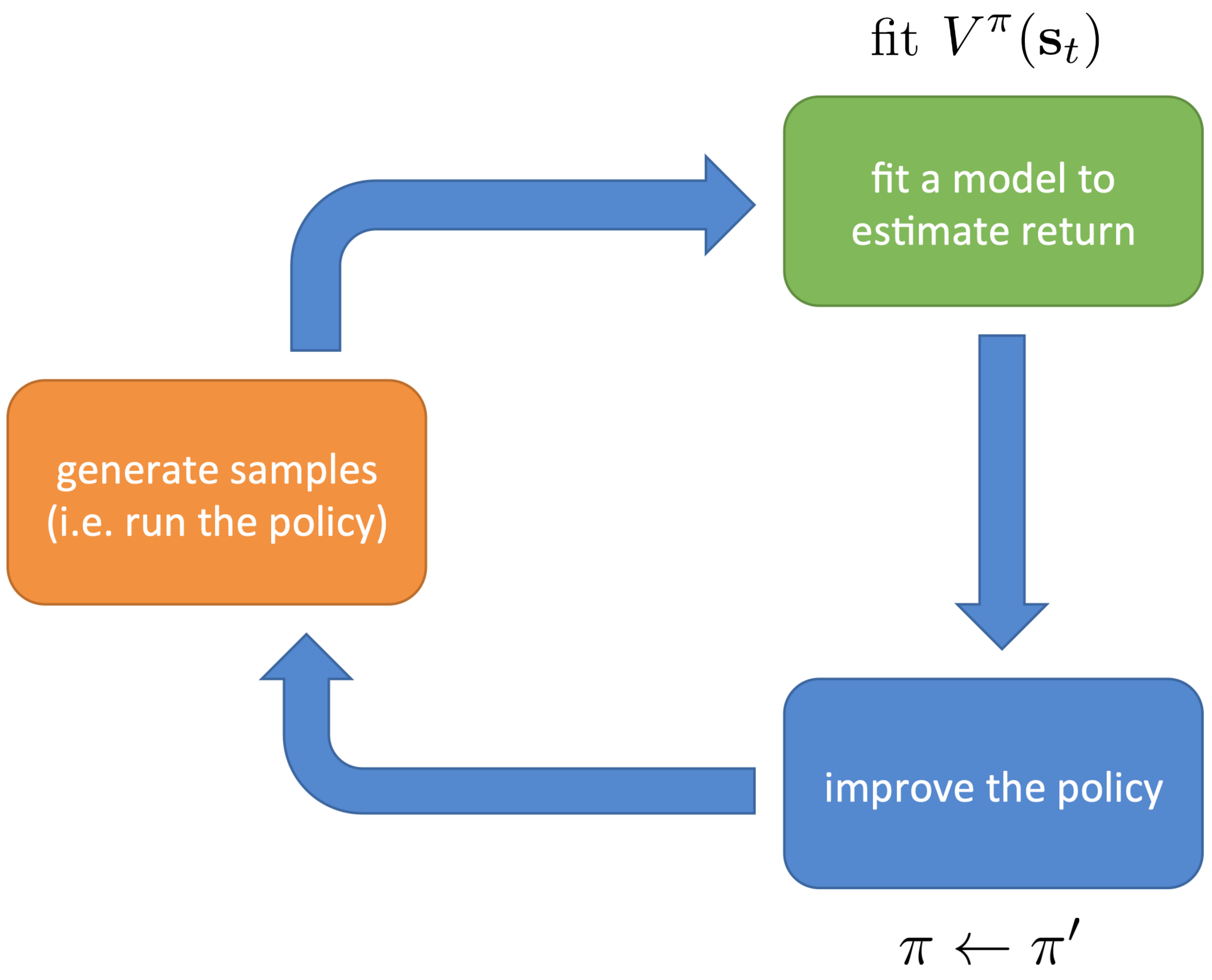
Policy Iteration

policy iteration algorithm:

1. evaluate $A^\pi(\mathbf{s}, \mathbf{a})$
2. set $\pi \leftarrow \pi'$

$$\pi'(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} A^\pi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 \text{ otherwise} \end{cases}$$

as before: $A^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma E[V^\pi(\mathbf{s}')] - V^\pi(\mathbf{s})$



Slide adapted from Sergey Levine

Value Iteration

policy iteration algorithm:

1. evaluate $Q^\pi(\mathbf{s}, \mathbf{a})$
2. set $\pi \leftarrow \pi'$

$$\pi'(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q^\pi(\mathbf{s}, \mathbf{a}) \\ 0 \text{ otherwise} \end{cases}$$

$$\arg\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a}) \rightarrow \text{policy}$$

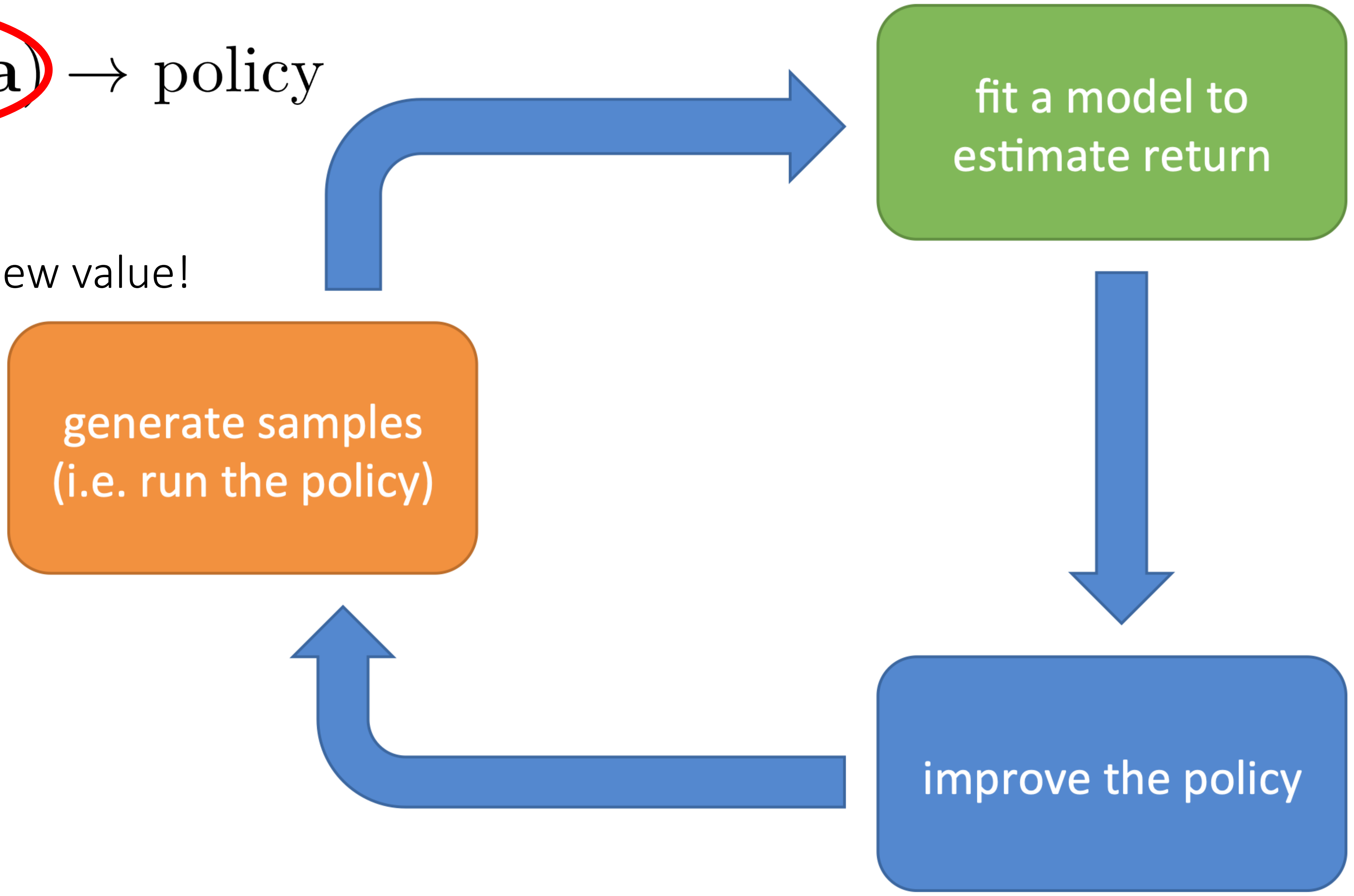
approximates the new value!

$$A^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma E[V^\pi(\mathbf{s}')] - V^\pi(\mathbf{s})$$

$$\arg\max_{\mathbf{a}_t} A^\pi(\mathbf{s}_t, \mathbf{a}_t) = \arg\max_{\mathbf{a}_t} Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$$

$$Q^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma E[V^\pi(\mathbf{s}')] \text{ (a bit simpler)}$$

$$Q^\pi(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})}[V^\pi(\mathbf{s}')]$$



$$V^\pi(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a})$$

skip the policy and compute values directly!

value iteration algorithm:

1. set $Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E[V(\mathbf{s}')]$
2. set $V(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$

The Plan

Advanced policy gradients recap

Actor-critic & case studies

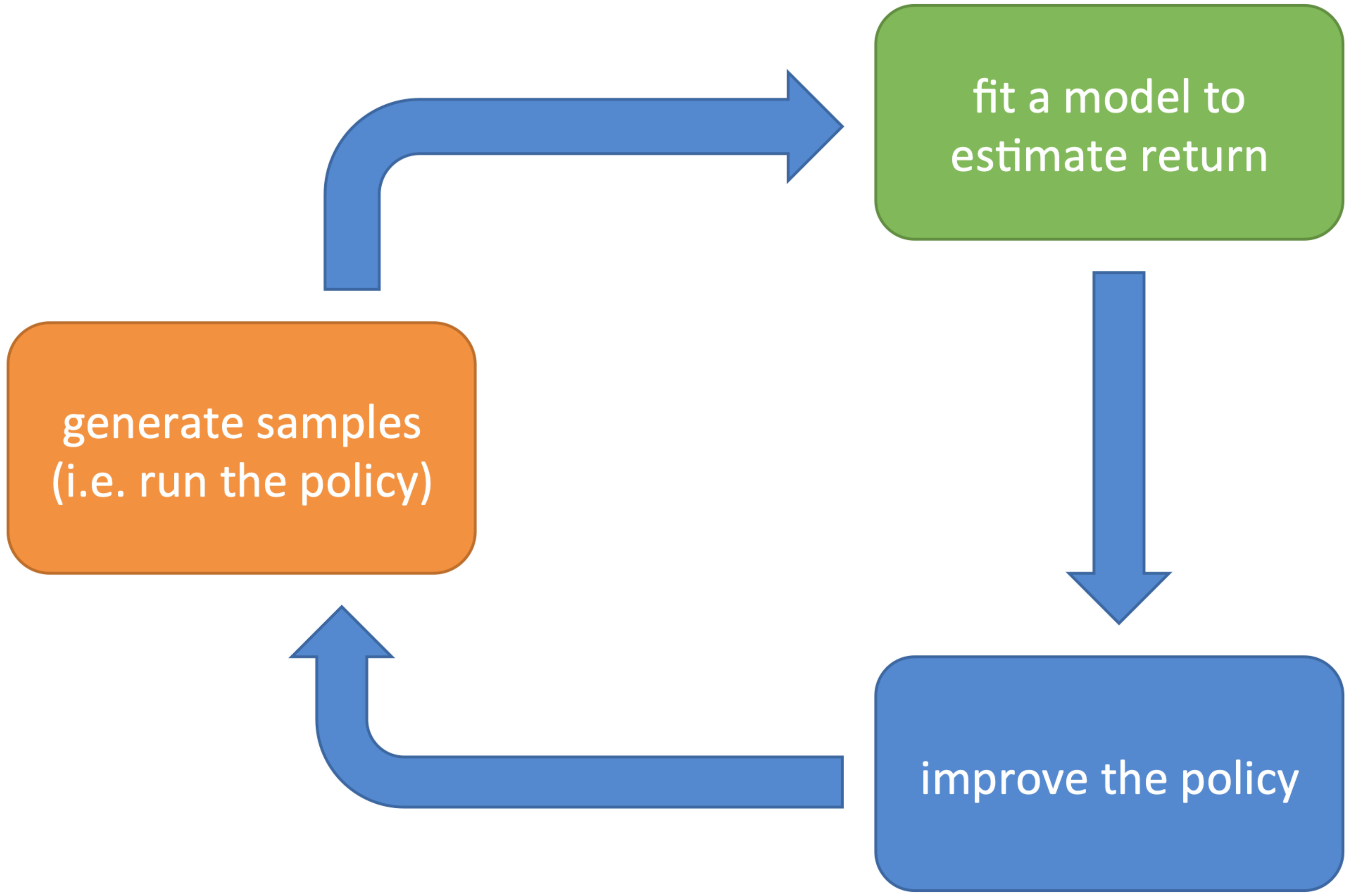
Policy iteration and value iteration

Q learning

Q learning

$$Q^\pi(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \gamma E_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}[V^\pi(\mathbf{s}')]$$

$$\pi'(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q^\pi(\mathbf{s},\mathbf{a}) \\ 0 \text{ otherwise} \end{cases}$$



$$V^\pi(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q^\pi(\mathbf{s},\mathbf{a})$$

value iteration algorithm:

1. set $Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \gamma E[V(\mathbf{s}')]$
2. set $V(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q(\mathbf{s},\mathbf{a})$

fitted Q iteration algorithm:

1. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i,\mathbf{a}_i) + \gamma E[V_\phi(\mathbf{s}'_i)]$ $\longleftarrow$ approxiate $E[V(\mathbf{s}'_i)] \approx \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i,\mathbf{a}'_i)$
2. set $\phi \leftarrow \arg\min_\phi \frac{1}{2}\sum_i \|Q_\phi(\mathbf{s}_i,\mathbf{a}_i) - \mathbf{y}_i\|^2$

doesn't require simulation of actions!

Value-Based RL: Definitions

$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t\right]$: total reward from taking $\mathbf{a}_t$ in $\mathbf{s}_t$ "how good is a state-action pair”

$V^\pi(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)}[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)]$: total reward from $\mathbf{s}_t$ "how good is a state”

If you know Q^π, you can use it to improve π.

$$\pi'(\mathbf{a}_t | \mathbf{s}_t) = \begin{cases} 1 \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} A^\pi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 \text{ otherwise} \end{cases}$$

For the optimal policy $\pi^\star$: $\underbrace{Q^*(\mathbf{s}_t, \mathbf{a}_t) = E_{\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)}\left[r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}'} Q^*(\mathbf{s}_{t+1}, \mathbf{a}')\right]}_{\text{Bellman equation}}$

Value-Based RL

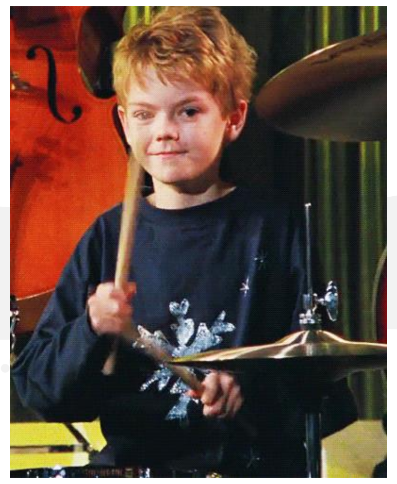
Value function: $V^{\pi}(\mathbf{s}_t) = ?$

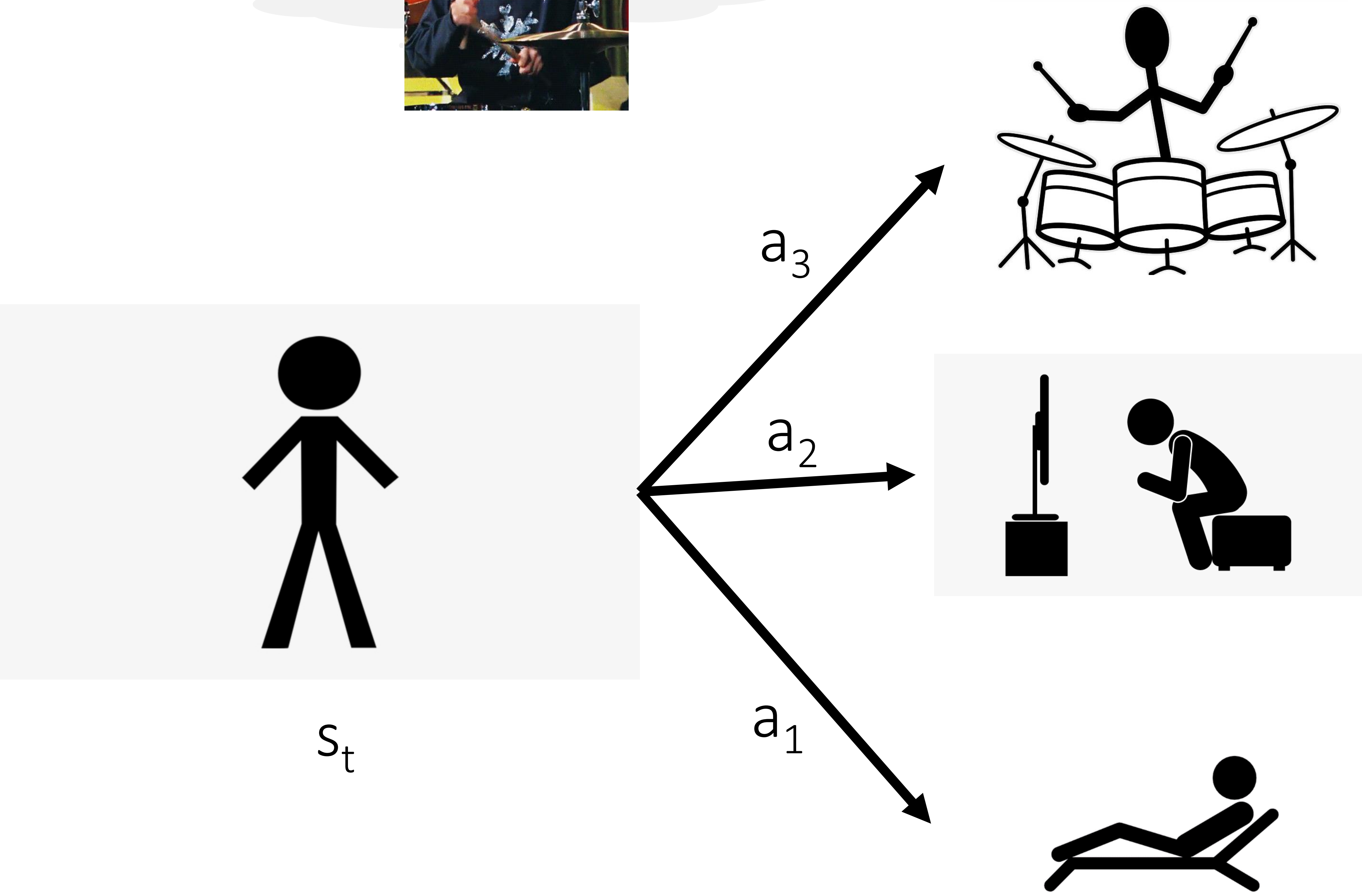
Q function: $Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = ?$

Q* function: $Q^{*}(\mathbf{s}_t, \mathbf{a}_t) = ?$

Value* function: $V^{*}(\mathbf{s}_t) = ?$

Reward = 1 if I can play it in a month, 0 otherwise





Current $\pi(\mathbf{a}_1|\mathbf{s}) = 1$

Fitted Q-iteration Algorithm

full fitted Q-iteration algorithm:

1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ <u>using some policy</u>

$K\times$

2. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. set $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$

Algorithm hyperparameters

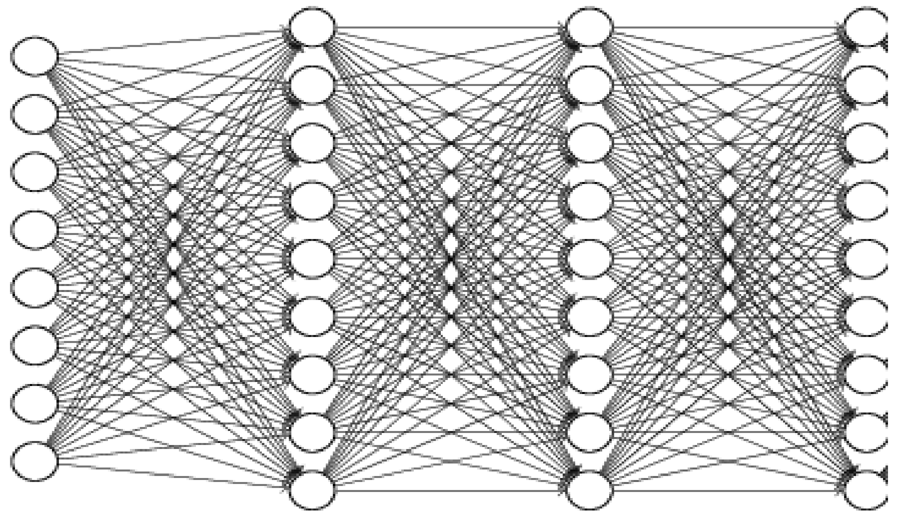
dataset size N, collection policy

iterations K

gradient steps S

$\mathbf{s}$

$\mathbf{a}$



$Q_\phi(\mathbf{s}, \mathbf{a})$

parameters ϕ

Result: get a policy $\pi(\mathbf{a}|\mathbf{s})$ from $\arg\max_{\mathbf{a}} Q_\phi(\mathbf{s}, \mathbf{a})$

Important notes:

We can reuse data from previous policies!

an off-policy algorithm using replay buffers

This is not a gradient descent algorithm!

Slide adapted from Sergey Levine

Q-learning

Bellman equation: $Q^*(\mathbf{s}_t, \mathbf{a}_t) = E_{\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}'} Q^*(\mathbf{s}_{t+1}, \mathbf{a}') \right]$

Pros:

+ More sample efficient than on-policy methods
+ Can incorporate off-policy data (including a fully offline setting)
+ Can updates the policy even without seeing the reward
+ Relatively easy to parallelize

Cons:

- Lots of "tricks" to make it work
- Potentially could be harder to learn than just a policy

Recap

Key learning goals:

- Understand the difference between policy & value iteration
- Intuition of Q learning

Policy & value iteration:

- Iterate either over the policy or the value function
- Value iteration -> Q-learning

Q learning:

- A different way to improve policy
- No explicit policy necessary
- Off-policy method

Next

How to implement Q learning in practice?

Can we improve it even more?

More case studies