Multi-Task and Goal-Conditioned Reinforcement Learning

CS 330

Reminders

Today:

Homework 2 due, Homework 3 out (more light-weight)

Mid-quarter survey

Office hours can be in person or virtual
(check the calendar)

The Plan

Recap & Q-learning

Multi-task imitation and policy gradients

Multi-task Q-learning

Goal-conditioned RL

The Plan

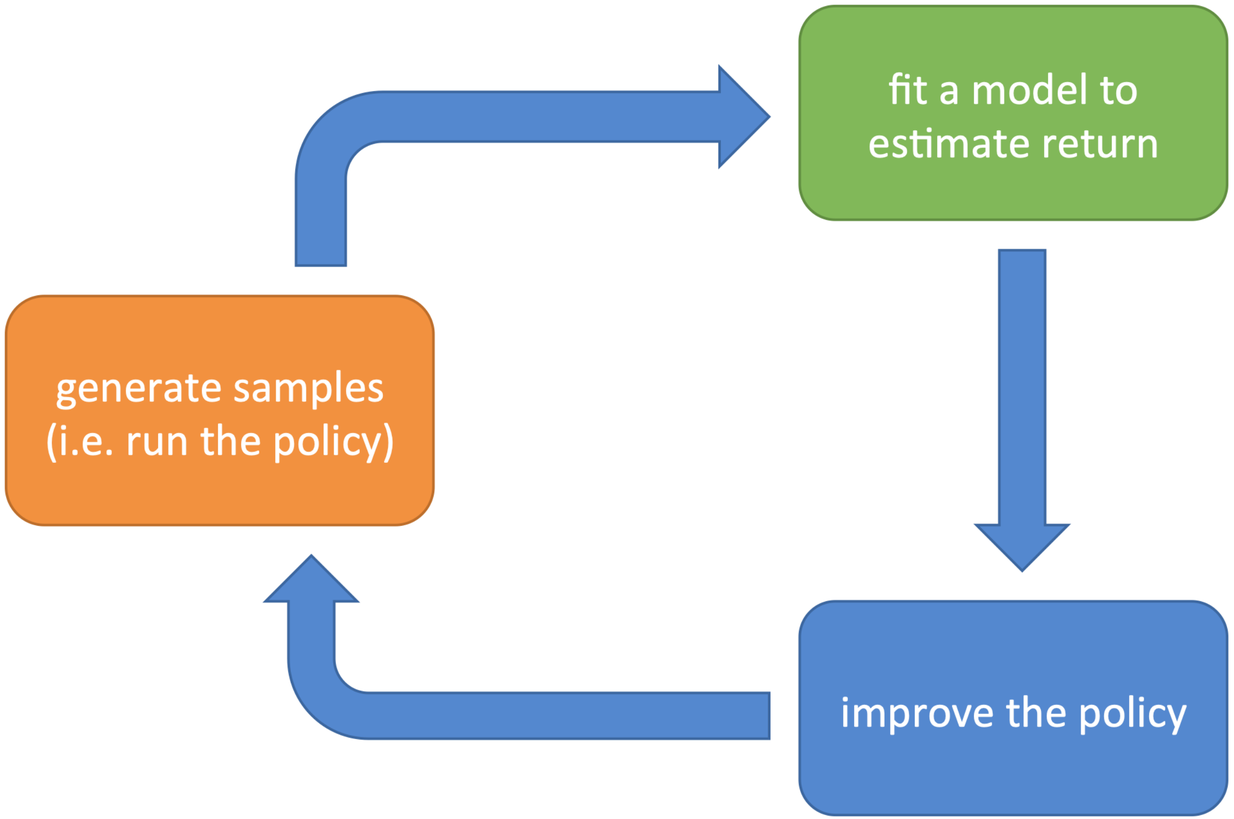
Recap & Q-learning

Multi-task imitation and policy gradients

Multi-task Q-learning

Goal-conditioned RL

The anatomy of a reinforcement learning algorithm



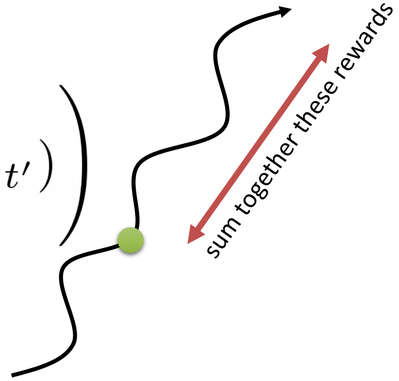
compute $\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ (MC policy gradient)
fit $Q_\phi(\mathbf{s}, \mathbf{a})$ (actor-critic, Q-learning)
estimate $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ (model-based)

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

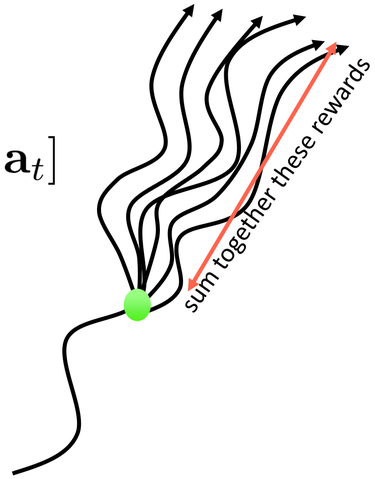
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \left(\sum_{t'=t}^{T} r(\mathbf{a}_{i,t'}, \mathbf{s}_{i,t'}) \right)$$

Multi-Step Prediction

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



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



- 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?

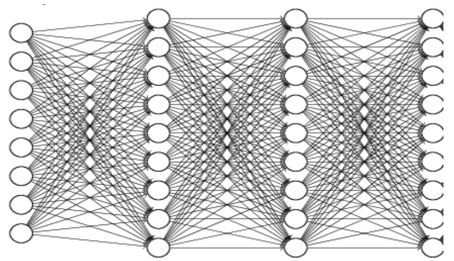
How can we use all of this to fit a better estimator?

Goal: fit V^π

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

directly use previous fitted value function!

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

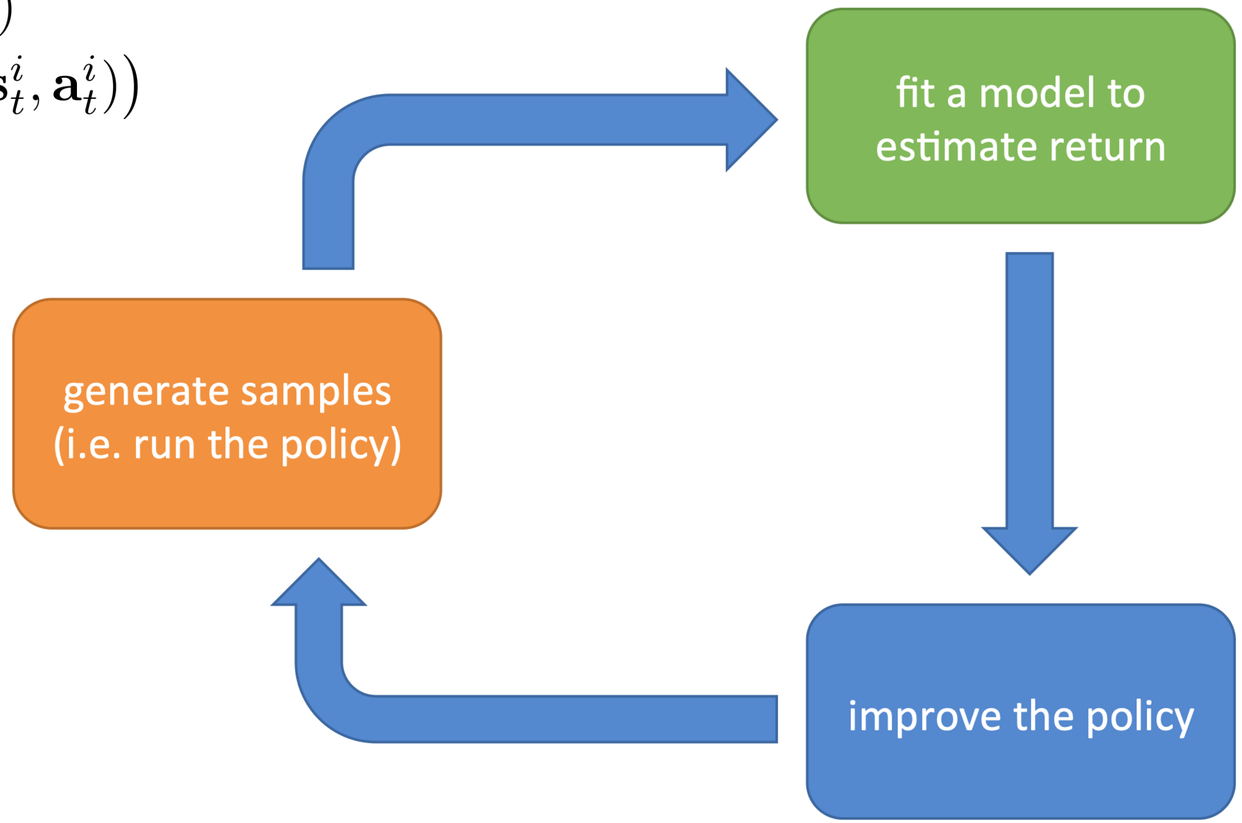
Slide adapted from Sergey Levine

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



Slide adapted from Sergey Levine

This was just the prediction part…

Improving the policy

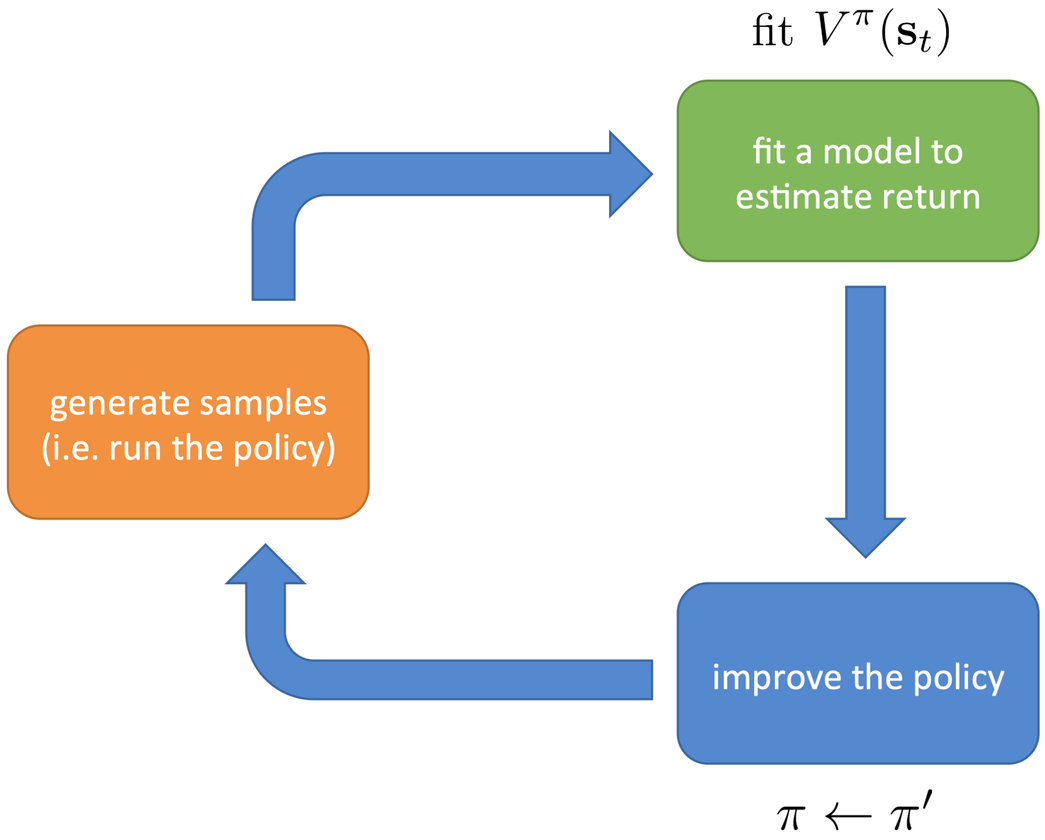
$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \text{ (policy gradient)}$$

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

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

how good is an action compared to the policy?

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



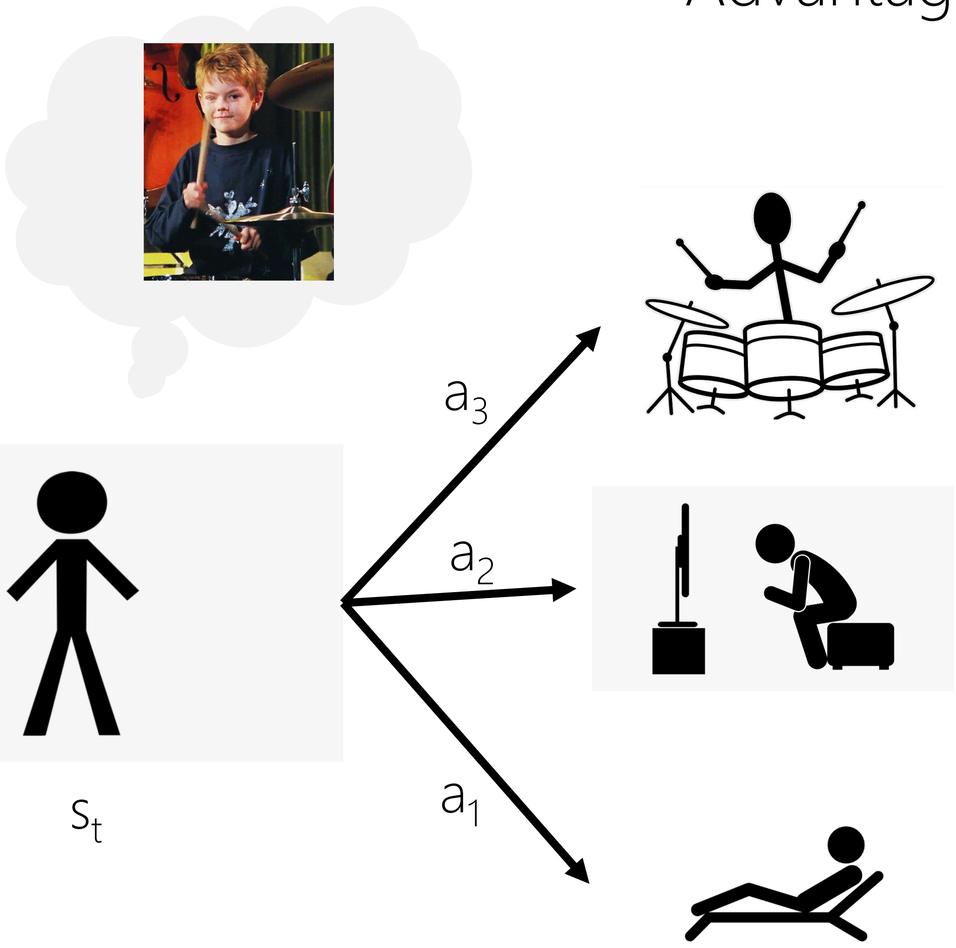
Value-based RL

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

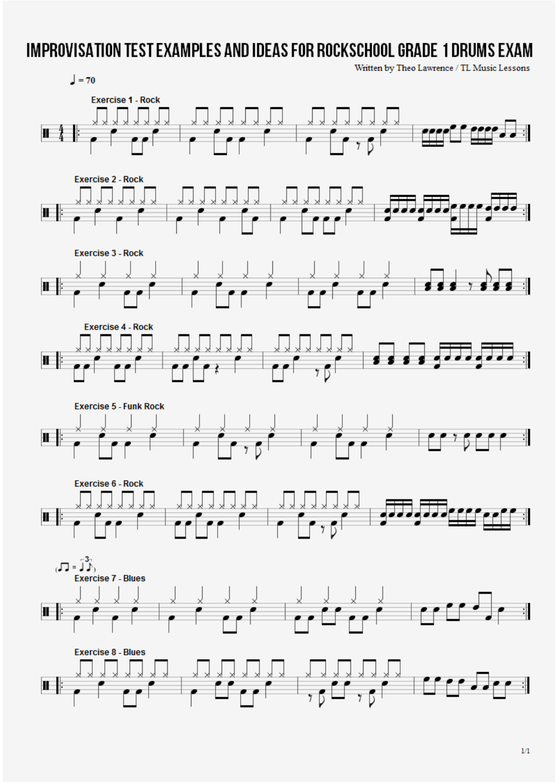
Q function: $Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = ?$

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

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



How can we improve the policy?



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

Improving the policy

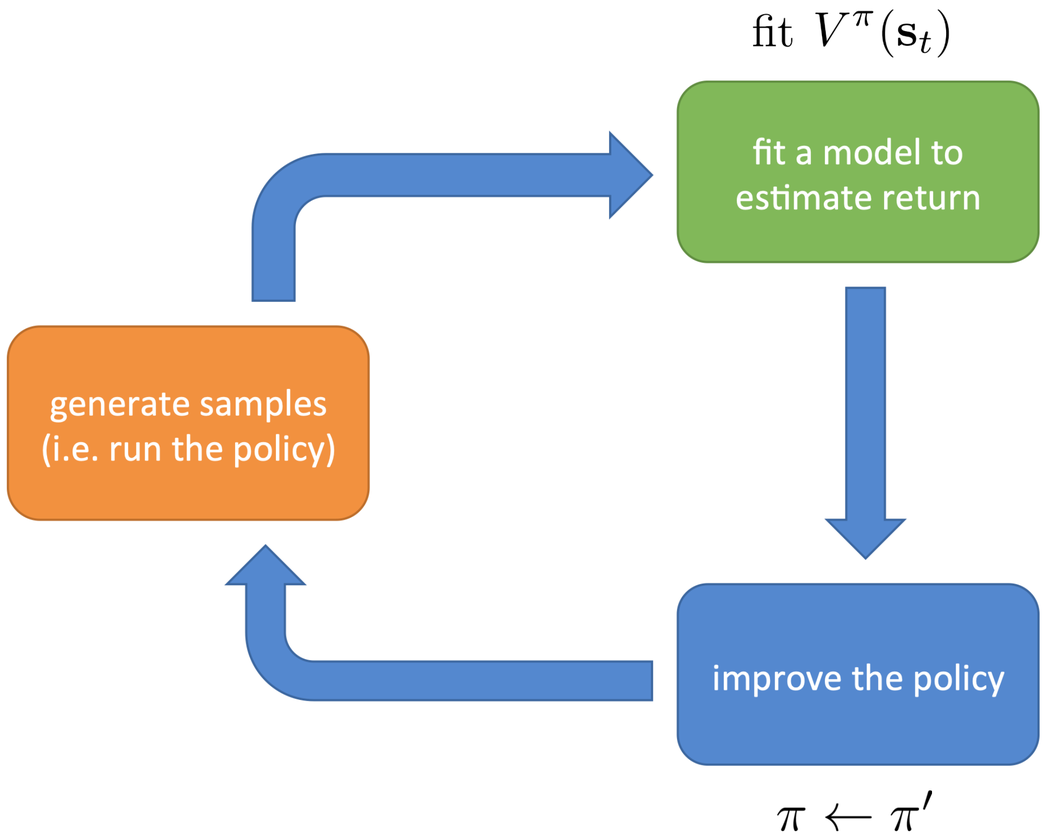
$A^\pi(\mathbf{s}_t, \mathbf{a}_t)$: how much better is $\mathbf{a}_t$ than the average action according to π

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

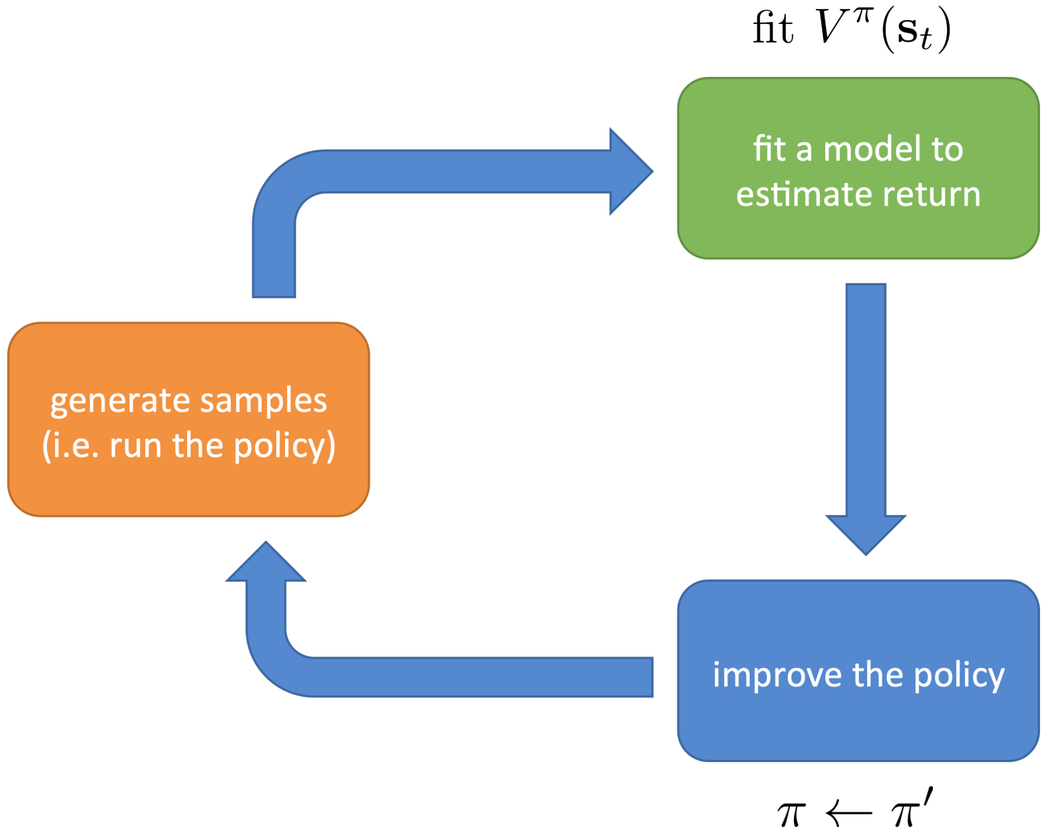
Policy iteration

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 $\pi \leftarrow \pi'$

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

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

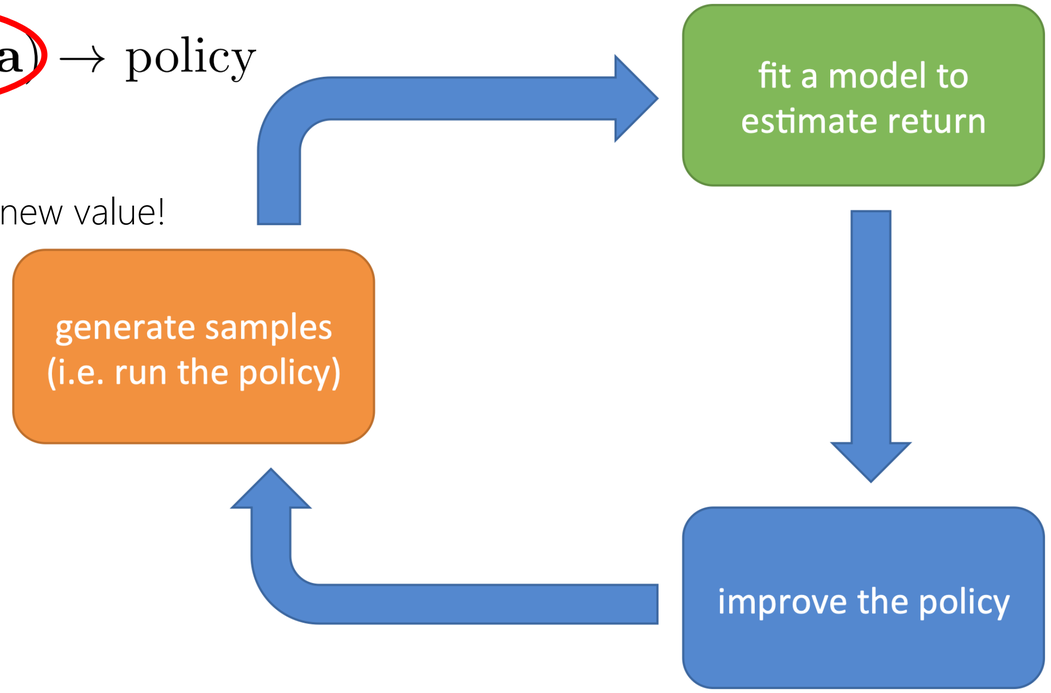
$\arg\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a}) \rightarrow$ policy

approximates the new value!

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

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

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



skip the policy and compute values directly!

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

value iteration algorithm:

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

Slide adapted from Sergey Levine

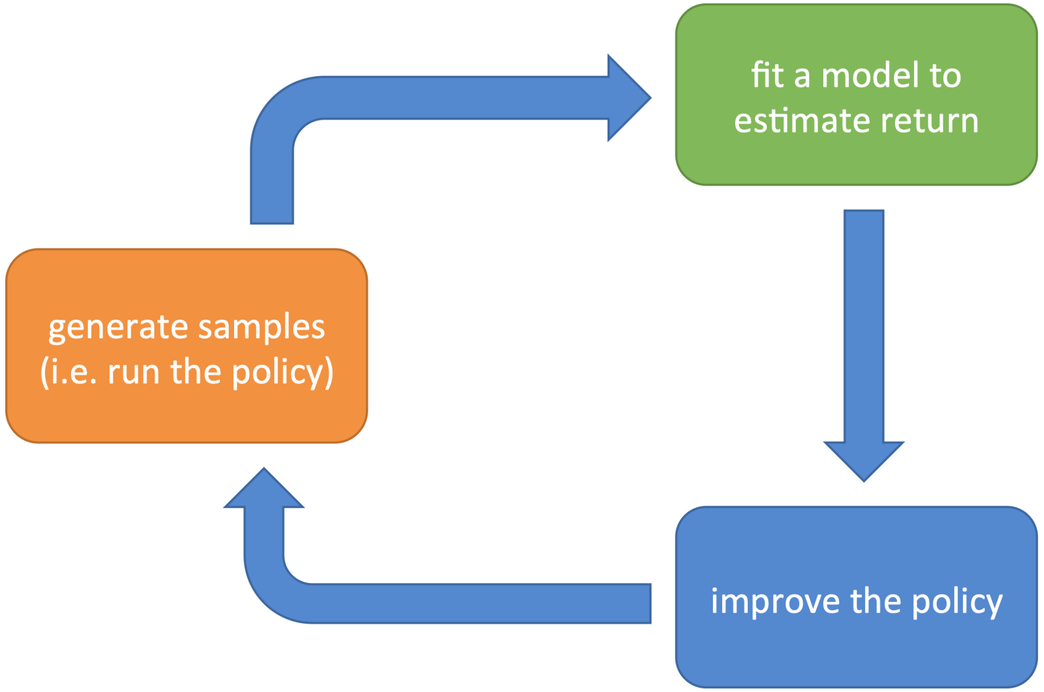
Q learning

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

value iteration algorithm:

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

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



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

fitted Q iteration algorithm:

1. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma E[V_\phi(\mathbf{s}'_i)]$ ← 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

Value function: $V^{\pi}(\mathbf{s}_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \mid \mathbf{s}_t\right]$ total reward starting from $\mathbf{s}$ and following π

"how good is a state"

Q function: $Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \mid \mathbf{s}_t, \mathbf{a}_t\right]$ total reward starting from $\mathbf{s}$, taking $\mathbf{a}$, and then following π

"how good is a state-action pair"

For the optimal policy $\pi^{\star}$: $Q^{\star}(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\mathbf{s}' \sim p(\cdot|\mathbf{s},\mathbf{a})}\left[r(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q^{\star}(\mathbf{s}', \mathbf{a}')\right]$

Bellman equation

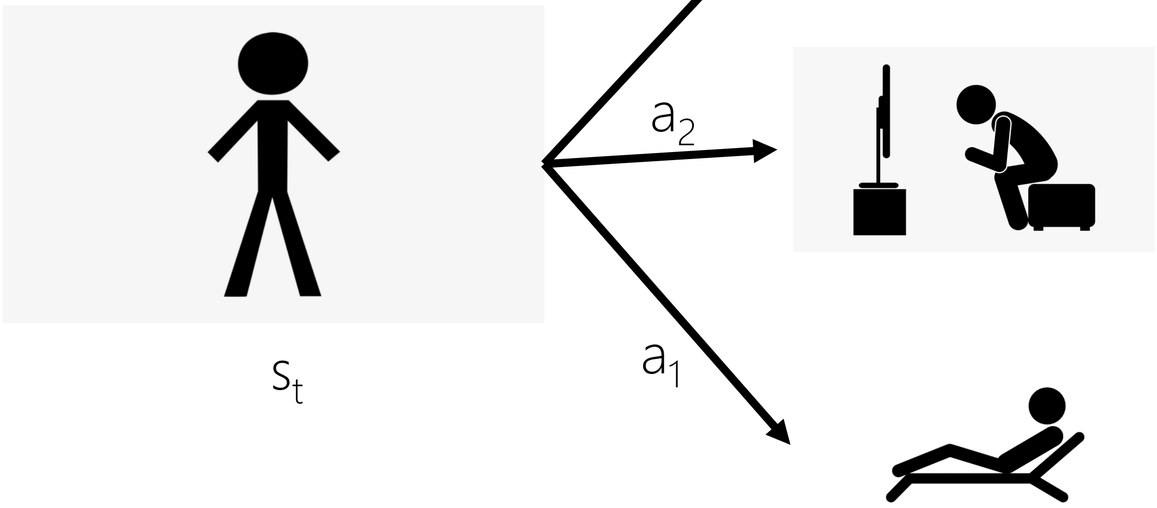
Value-based RL

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

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

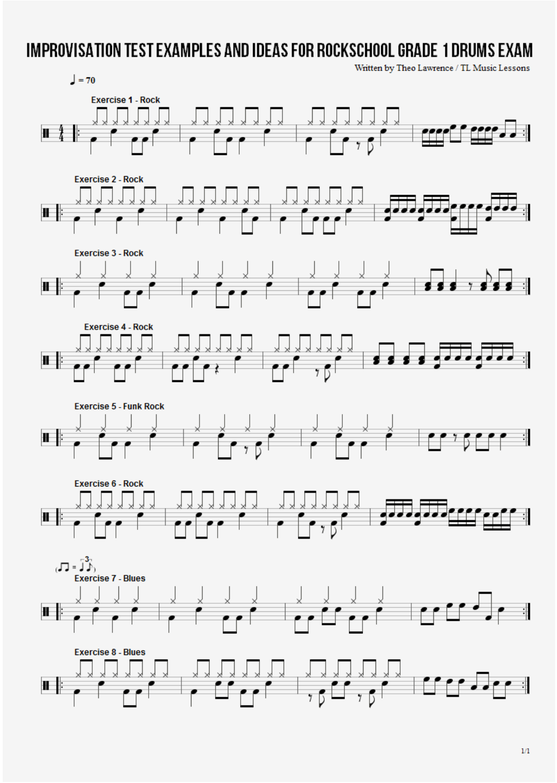
Q* function: $Q^{*}(\mathbf{s}_t, \mathbf{a}_t) = ?$

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



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

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



Fitted Q-iteration algorithm

full fitted Q-iteration algorithm:

$K\times$

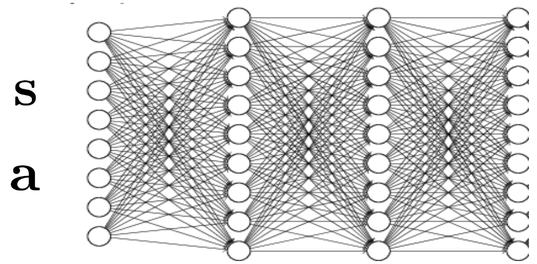
1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy
2. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. set $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \left\| Q_\phi(\mathbf{s}_i,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

Algorithm hyperparameters

dataset size N, collection policy

iterations K

gradient steps S



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

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

Important notes:

We can reuse data from previous policies!
an off-policy algorithm using replay buffers

This is not a gradient descent algorithm!

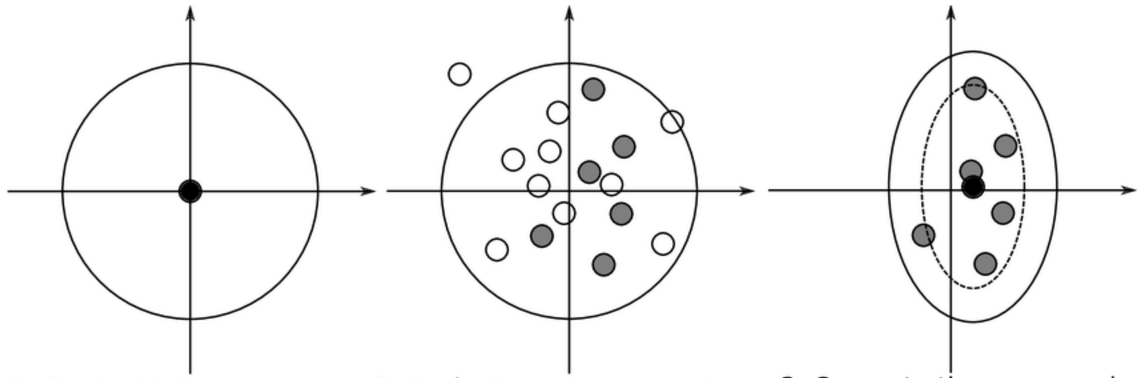
Slide adapted from Sergey Levine

Example: Q-learning applied to robotics

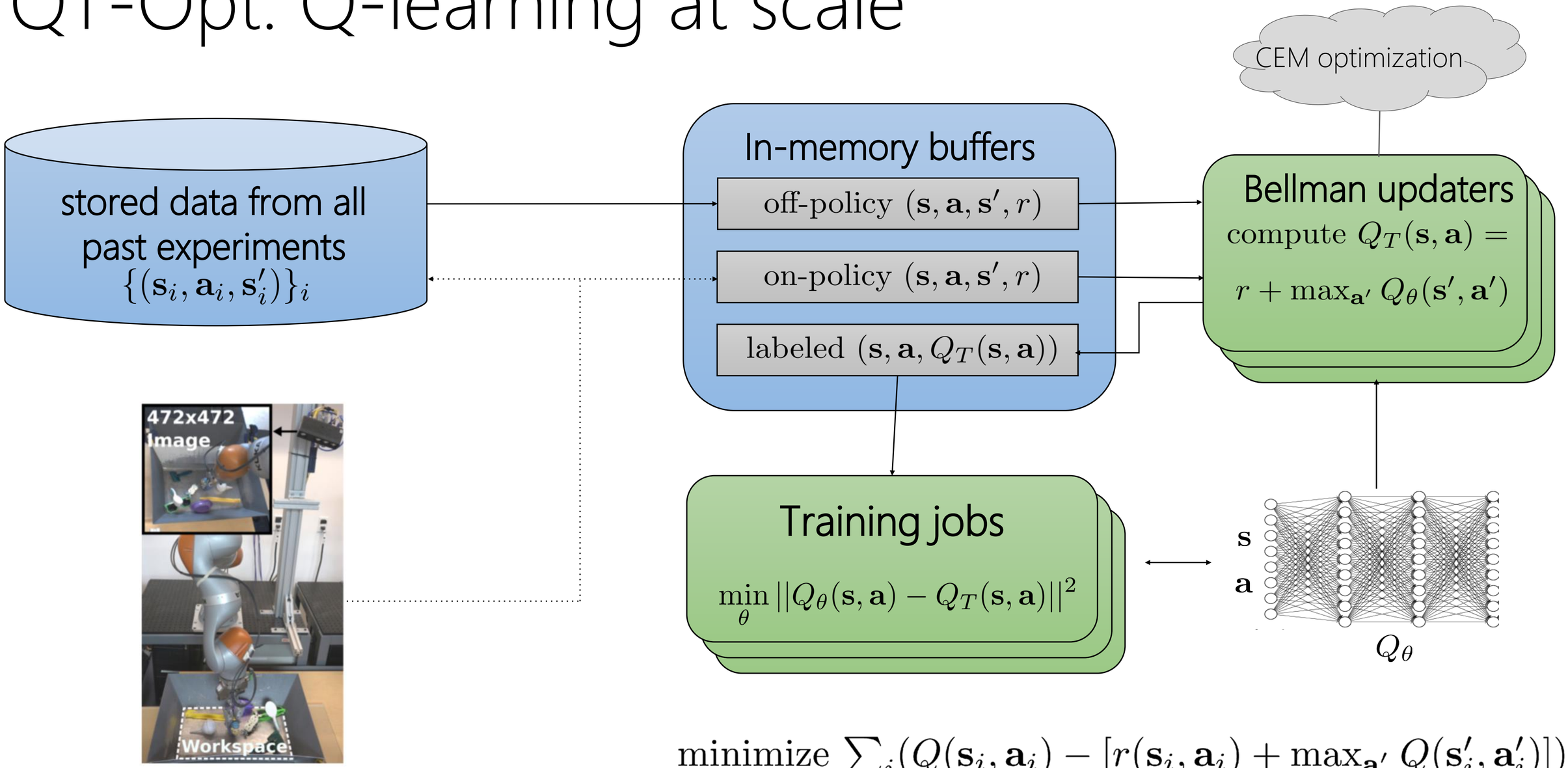
1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy
2. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. set $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i,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

Continuous action space?

Simple optimization algorithm
Cross Entropy Method (CEM)



QT-Opt: Q-learning at scale

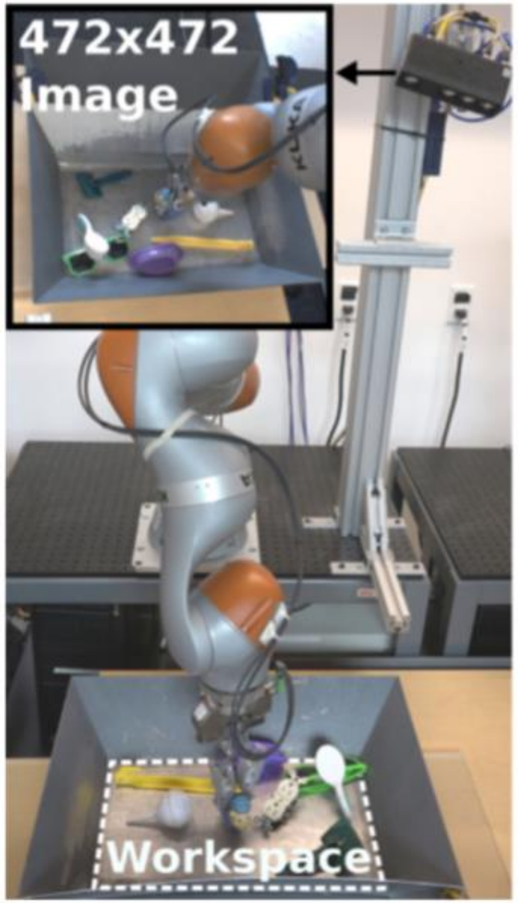


$$\text{minimize} \sum_i (Q(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \max_{\mathbf{a}'_i} Q(\mathbf{s}'_i, \mathbf{a}'_i)])^2$$

Slide adapted from D. Kalashnikov

QT-Opt: Kalashnikov et al. '18, Google Brain

QT-Opt: MDP definition for grasping

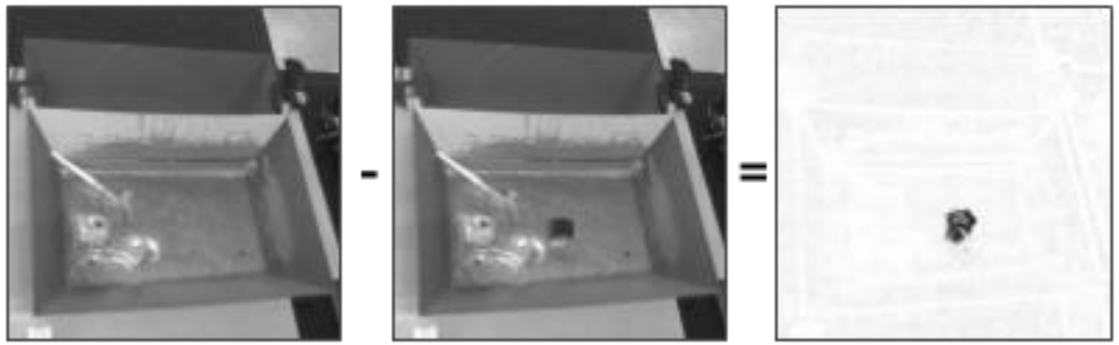


State: over the shoulder RGB camera image, no depth

Action: 4DOF pose change in Cartesian space + gripper control

Reward: binary reward at the end, if the object was lifted. Sparse. No shaping

Automatic success detection:



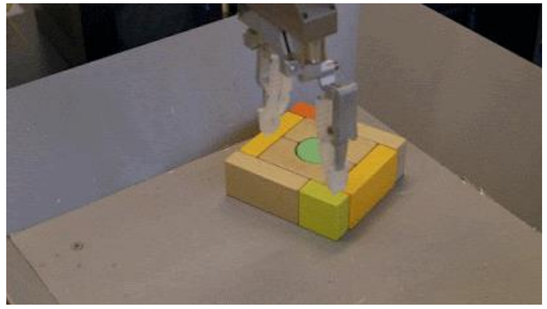
Slide adapted from D. Kalashnikov

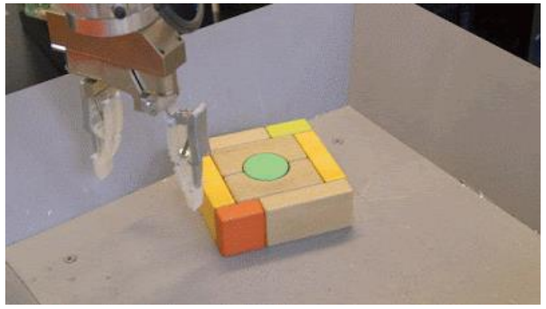
QT-Opt: setup and results

7 robots collected 580k grasps

Unseen test objects

96% test success rate!





Q-learning

Bellman equation: $Q^\star(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\mathbf{s}' \sim p(\cdot|\mathbf{s},\mathbf{a})}\left[r(\mathbf{s},\mathbf{a}) + \gamma \max_{\mathbf{a}'} Q^\star(\mathbf{s}', \mathbf{a}')\right]$

Pros:

+ More sample efficient than on-policy methods
+ Can incorporate off-policy data (including a fully offline setting)
+ Can update 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

The Plan

Recap & Q-learning

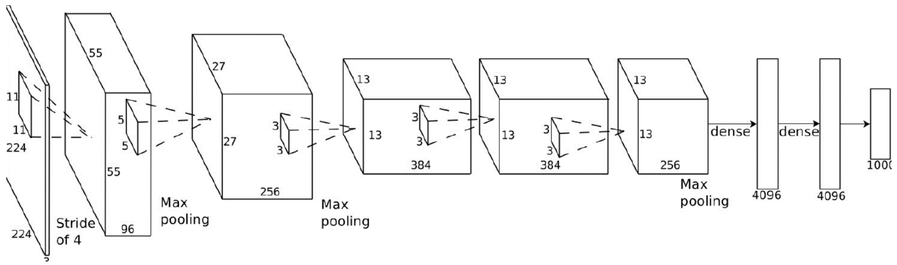
Multi-task imitation and policy gradients

Multi-task Q-learning

Goal-conditioned RL

Multi-task imitation learning

$\mathbf{o}_t$

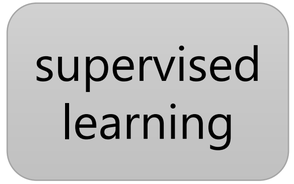


$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

$\mathbf{a}_t$

$\mathbf{o}_t$

$\mathbf{a}_t$



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

Images: Bojarski et al. ‘16, NVIDIA

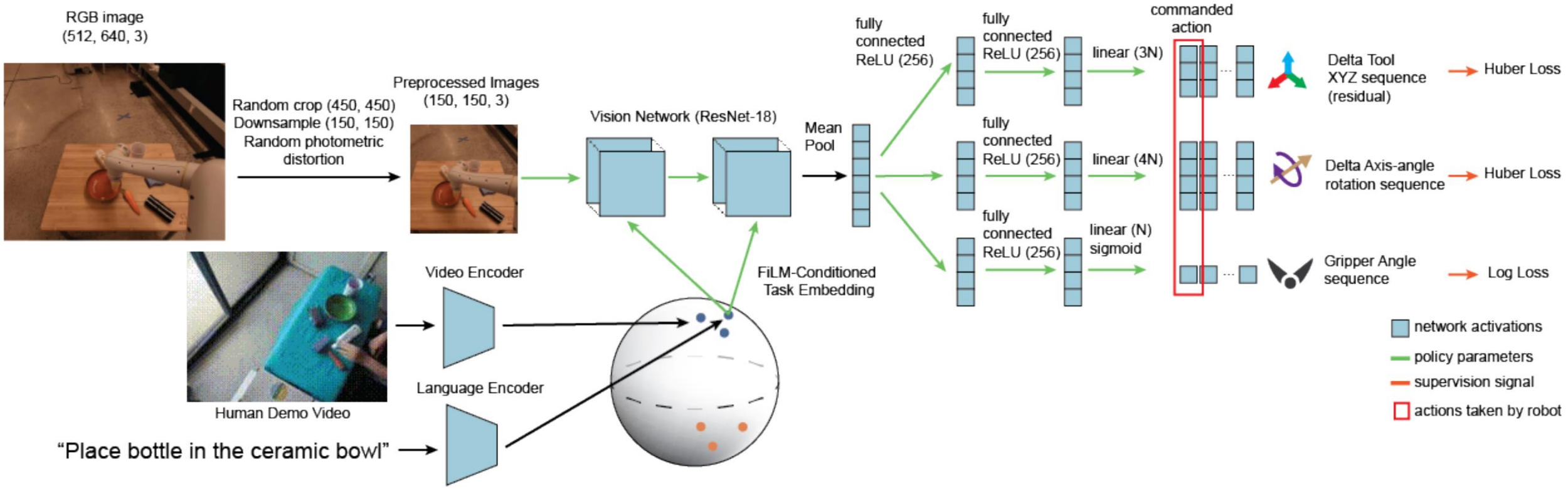
How to optimize multi-task IL?

Reminder (lecture 2): Vanilla MTL Objective: $\min_{\theta} \sum_{i=1}^{T} \mathscr{L}_i(\theta, \mathscr{D}_i)$

Same as supervised learning!

Same architectures, stratified sampling, soft/hard weight sharing

How to specify a task?



$$\min \sum_{\text{task } i} \sum_{\substack{(s,a)\sim\mathcal{D}_e^i \\ w_h\sim\mathcal{D}_h^i \bigcup \mathcal{D}_e^i}} \underbrace{-\log \pi(a|s, z^i)}_{\text{behavior cloning}} + \underbrace{D_{\cos}(z_h^i, z_\ell^i)}_{\text{language regression}} \text{, where } \underbrace{z_h^i \sim q(\cdot|w_h)}_{\text{video encoder}}, \underbrace{z_\ell^i \sim q(\cdot|w_\ell^i)}_{\text{language encoder}}$$

Jang et al. BC-Z. CoRL 2021

How to specify a task?

Skill	Held-out tasks (no demos during training)	Lang-conditioned performance
pick-place	‘place sponge in tray’	82% (9.2)
	‘place grapes in red bowl’	75% (10.8)
	‘place apple in paper cup’	33% (12.2)
pick-wipe	‘wipe tray with sponge’	0% (0)
pick-place	‘place banana in ceramic bowl’	75% (9.7)
	‘place bottle in red bowl’	75% (9.7)
	‘place grapes in ceramic bowl’	70% (10.3)
	‘place bottle in table surface’	50% (11.2)
	‘place white sponge in purple bowl’	45% (11.2)
	‘place white sponge in tray’	40% (11.0)
	‘place apple in ceramic bowl’	20% (8.9)
	‘place bottle in purple bowl’	20% (8.9)
	‘place banana in ceramic cup’	0% (0)
	‘place banana on white sponge’	0% (0)
	‘place metal cup in red bowl’	0% (0)
grasp	‘pick up grapes’	65% (10.7)
	‘pick up apple’	55% (11.2)
	‘pick up towel’	42.8% (18.7)
	‘pick up pepper’	35% (10.7)
	‘pick up bottle’	30% (10.3)
	‘pick up the red bowl’	0% (0)
pick-drag	‘drag grapes across the table’	14% (13.2)
pick-wipe	‘wipe table surface with banana’	10% (6.7)
	‘wipe tray with white sponge’	0% (0)
	‘wipe ceramic bowl with brush’	0% (0)
push	‘push purple bowl across the table’	30% (10.3)
	‘push tray across the table’	25% (9.7)
	‘push red bowl across the table’	0% (0)
	Holdout Task Overall	32%

Shows non-zero success for 20/28 hold-out tasks

Average 32% success over all 28 tasks

“Push purple bowl across the table”

“Place bottle in tray”

Jang et al. BC-Z. CoRL 2021

What is a reinforcement learning task?

Reinforcement learning

A task: $\mathcal{T}_i \triangleq \{\mathcal{S}_i, \mathcal{A}_i, p_i(\mathbf{s}_1), p_i(\mathbf{s}'|\mathbf{s},\mathbf{a}), r_i(\mathbf{s},\mathbf{a})\}$

action space

dynamics

state space

initial state distribution

reward

An alternative view:

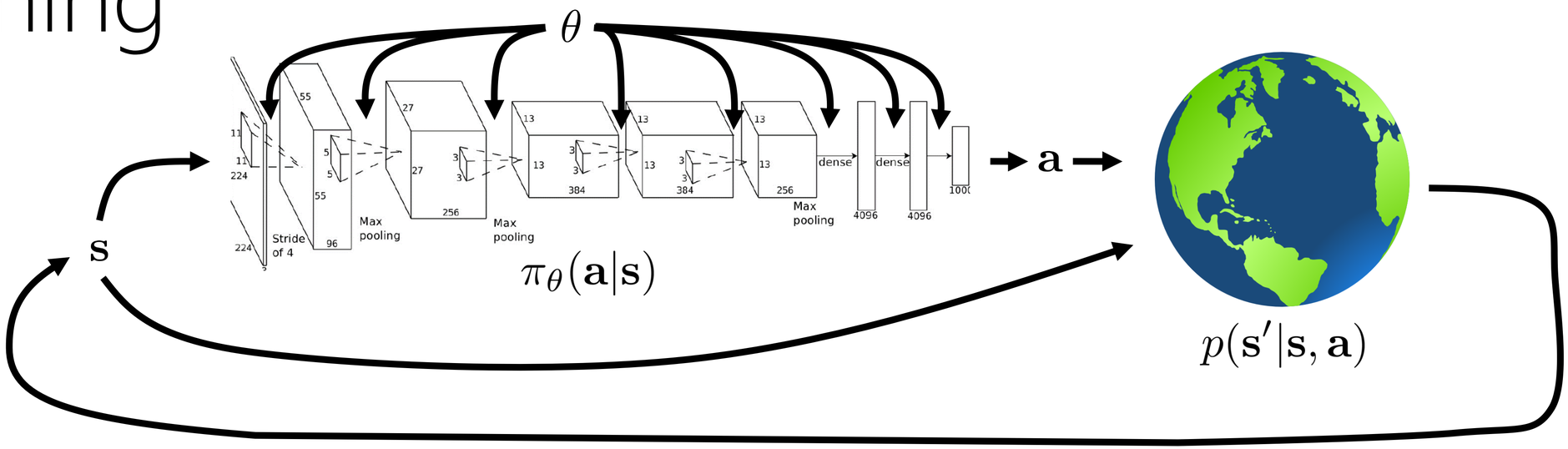
A task identifier is part of the state: $\mathbf{s} = (\overline{\mathbf{s}}, \mathbf{z}_i)$

original state

$$\mathcal{T}_i \triangleq \{\mathcal{S}_i, \mathcal{A}_i, p_i(\mathbf{s}_1), p(\mathbf{s}'|\mathbf{s},\mathbf{a}), r(\mathbf{s},\mathbf{a})\} \longrightarrow \{\mathcal{T}_i\} = \left\{\cup\mathcal{S}_i, \cup\mathcal{A}_i, \frac{1}{N}\sum_i p_i(\mathbf{s}_1), p(\mathbf{s}'|\mathbf{s},\mathbf{a}), r(\mathbf{s},\mathbf{a})\right\}$$

It can be cast as a standard Markov decision process!

The goal of multi-task reinforcement learning



Multi-task RL

The same as before, except:

a task identifier is part of the state: $\mathbf{s} = (\overline{\mathbf{s}}, \mathbf{z}_i)$

e.g. one-hot task ID

language description

desired goal state, $\mathbf{z}_i = \mathbf{s}_g$ ← "goal-conditioned RL"

What is the reward?

The same as before

Or, for goal-conditioned RL:

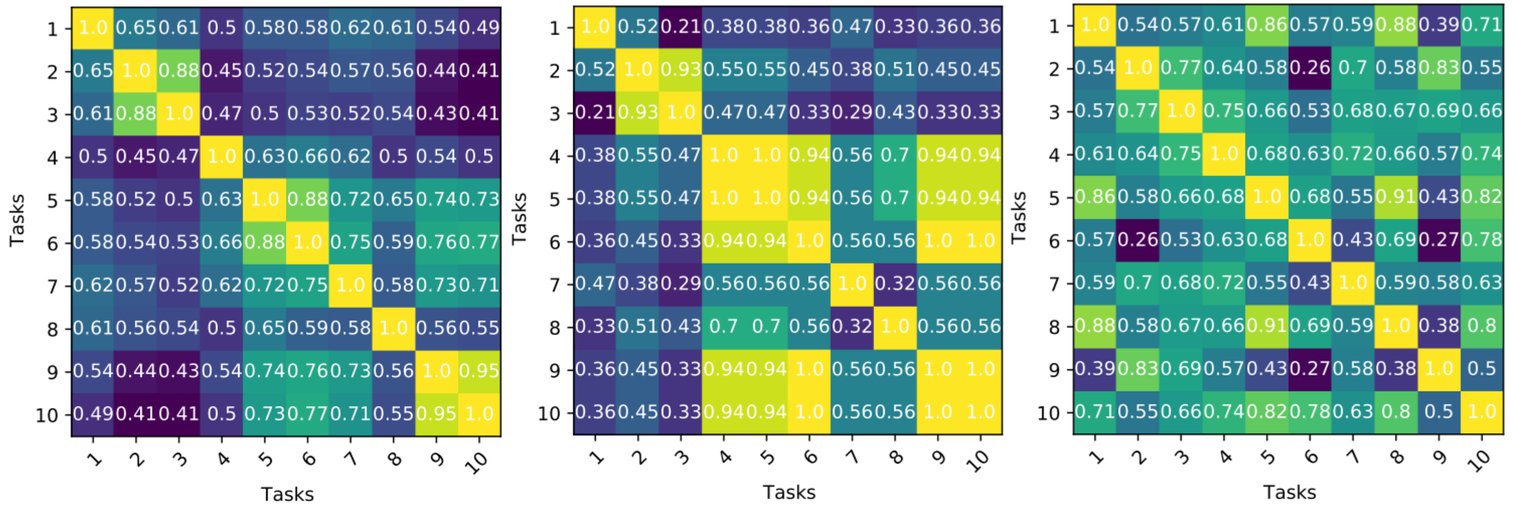
$r(\mathbf{s}) = r(\overline{\mathbf{s}}, \mathbf{s}_g) = -d(\overline{\mathbf{s}}, \mathbf{s}_g)$

Distance function d examples:

- Euclidean ℓ_2
- sparse 0/1

Context matters

1. Reach a goal position.
2. Push the puck to a goal.
3. Pick and place a puck to a goal.
4. Open a door with a revolving joint.
5. Open a drawer.
6. Push and close a drawer.
7. Press a button from the top.
8. Insert a peg sideways.
9. Push and open a window.
10. Push and close a window.



(a) (b) (c) (d)

Policy
Algorithm

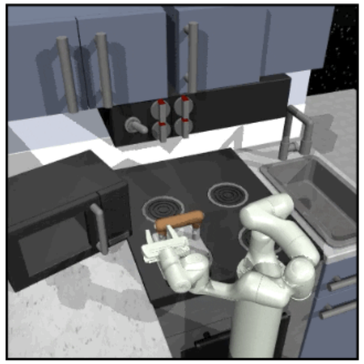
Figure 4. (a): Task descriptions for MT10. (b) Cosine similarity (for MT10) between the pretrained task embeddings. (c) Cosine similarity (for MT10) between the context representations using CARE model with $k = 6$ encoders. (d) Cosine similarity (for MT10) between the context representations without using metadata with $k = 6$ encoders. Structure across tasks is clearly exhibited in (b) and (c), but not in (d), which has no access to metadata.

Multi-task SAC (Yu et al., 2020b) *	0.36 ± 0.013
Multi-task SAC + Task Encoder *	0.40 ± 0.024
Multi-headed SAC (Yu et al., 2020b)	0.45 ± 0.064
PCGrad (Yu et al., 2020a)	0.5 ± 0.017
Soft Modularization (Yang et al., 2020)	0.5 ± 0.035
SAC + FiLM (Perez et al., 2018) *	0.40 ± 0.012
SAC + Metadata + ME (**CARE**)	$\mathbf{0.54 \pm 0.031}$

Multi-task (RL) benefits

Cross-task generalization

Easier exploration



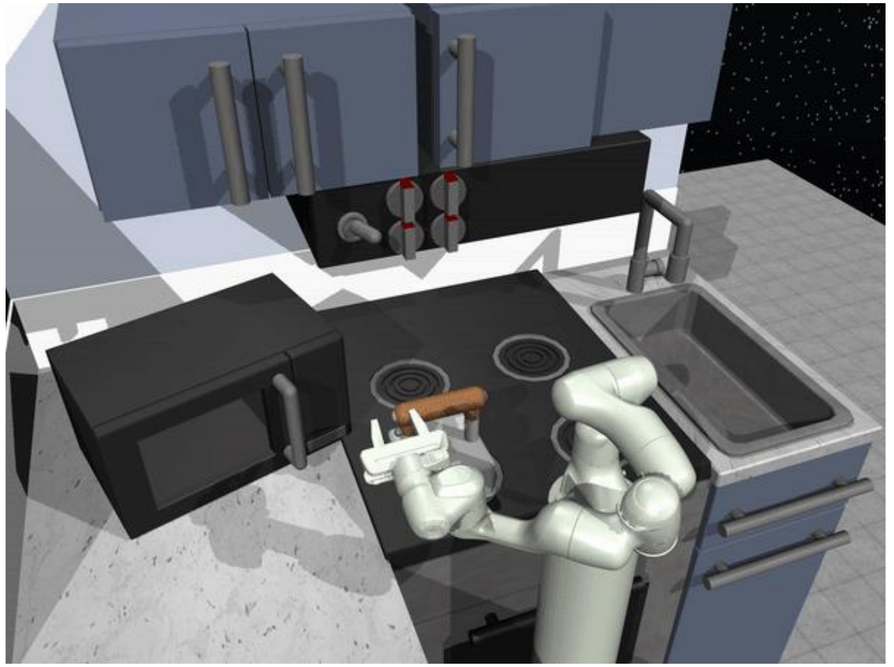
Pertsch et al. SPiRL

Multi-task (RL) benefits

Cross-task generalization

Easier exploration

Sequencing for long-horizon tasks



Gupta et al. Relay Policy Learning

Multi-task (RL) benefits

Cross-task generalization

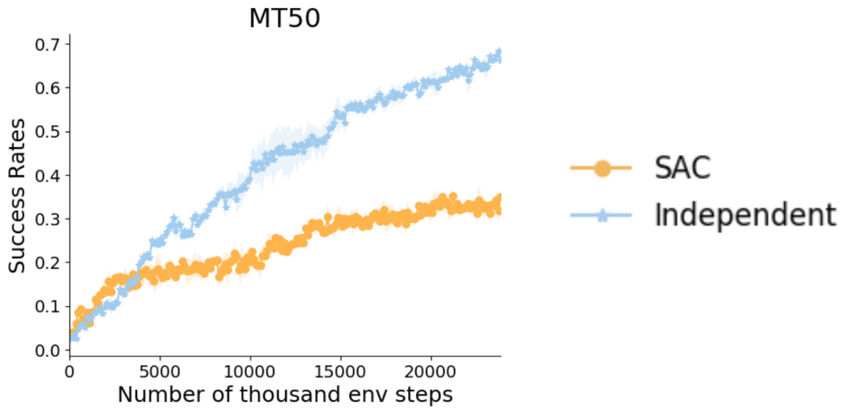
Easier exploration

Sequencing for long-horizon tasks

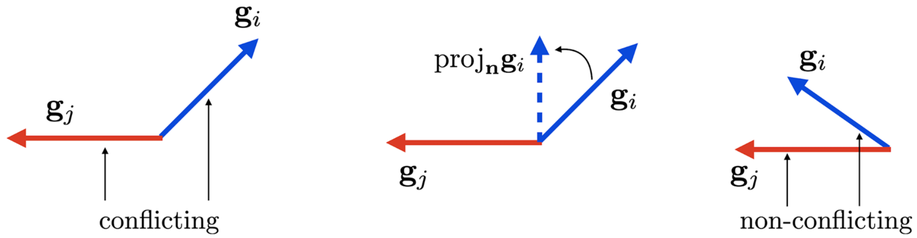
Reset-free learning

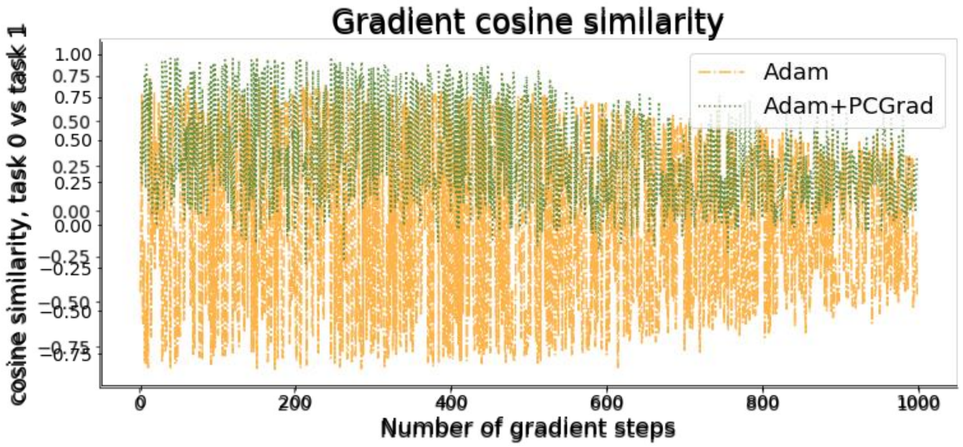
Per-task sample-efficiency gains

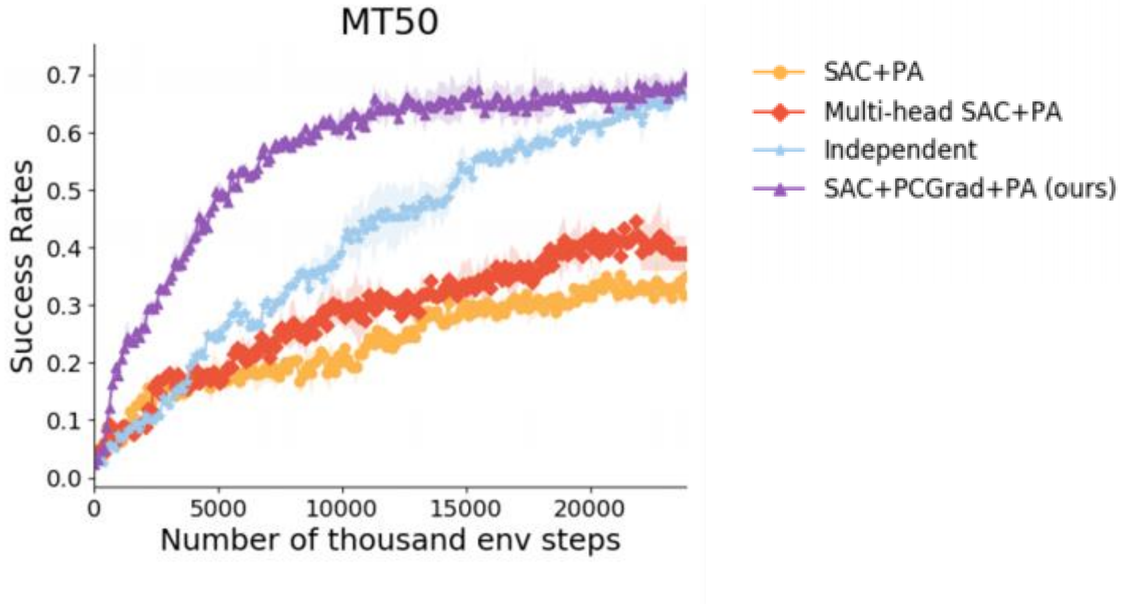
Multi-task (RL) difficulties



if two gradients conflict, project each onto the normal plane of the other, else, don't do anything.







Yu et al. PCGrad. NeurIPS '20

Multi-task RL algorithms

Policy: $\pi_\theta(\mathbf{a}|\bar{\mathbf{s}}) \longrightarrow \pi_\theta(\mathbf{a}|\bar{\mathbf{s}}, \mathbf{z}_i)$

Q-function: $Q_\phi(\bar{\mathbf{s}}, \mathbf{a}) \longrightarrow Q_\phi(\bar{\mathbf{s}}, \mathbf{a}, \mathbf{z}_i)$

Analogous to multi-task supervised learning: stratified sampling, soft/hard weight sharing, etc.

If it's still a standard Markov decision process,

then, why not apply standard RL algorithms? You can! You can often do better.

What is different about reinforcement learning?

The data distribution is
controlled by the agent!

Should we share data in addition to sharing weights?

The Plan

Recap & Q-learning

Multi-task imitation and policy gradients

Multi-task Q-learning

Goal-conditioned RL

An example

Task 1: passing

Task 2: shooting goals

What if you accidentally perform a good pass when trying to shoot a goal?

Store experience as normal. *and* Relabel experience with task 2 ID & reward and store.

“hindsight relabeling” "hindsight experience replay” (HER)

Multi-task RL with relabeling

1. Collect data $\mathcal{D}_k = \{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}, \mathbf{z}_i, r_{1:T})\}$ using some policy
2. Store data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k$
3. Perform hindsight relabeling:
   a. Relabel experience in $\mathcal{D}_k$ for task $\mathcal{T}_j$:
      $\mathcal{D}'_k = \{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}, \mathbf{z}_j, r'_{1:T}\}$ where $r'_t = r_j(\mathbf{s}_t)$
   b. Store relabeled data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}'_k$
4. Update policy using replay buffer $\mathcal{D}$

$k++$

<— Which task $\mathcal{T}_j$ to choose?

- randomly
- task(s) in which the trajectory gets high reward
- other

When can we apply relabeling?

- reward function form is known, evaluatable
- dynamics consistent across goals/tasks
- using an off-policy algorithm*

Eysenbach et al. Rewriting History with Inverse RL
Li et al. Generalized Hindsight for RL
Kalashnikov et al. MT-Opt
Yu et al. Conservative Data-Sharing

Kaelbling. Learning to Achieve Goals. IJCAI ‘93
Andrychowicz et al. Hindsight Experience Replay. NeurIPS ‘17

Another example:

Task 1: close a drawer

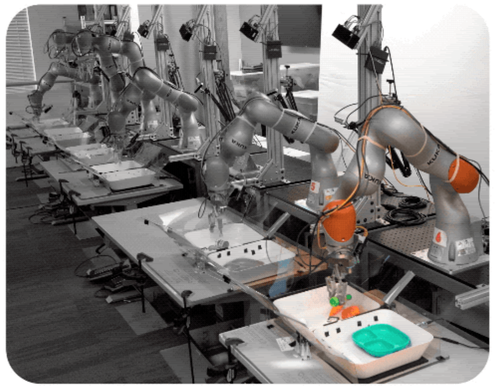
Task 2: open a drawer

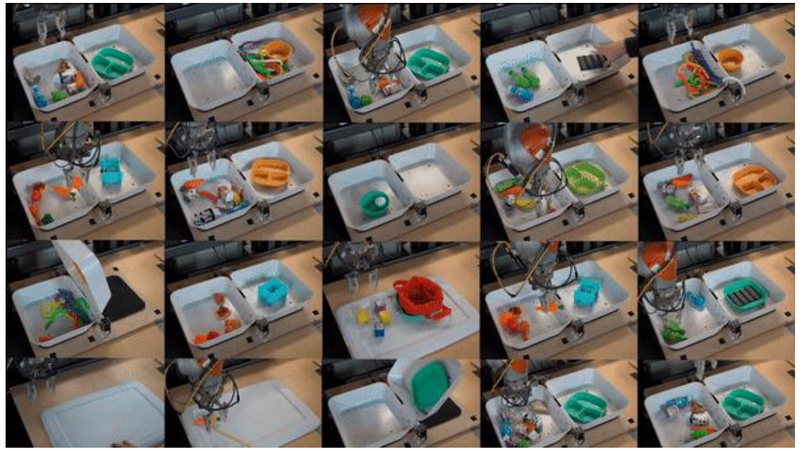
Can we use episodes from drawer opening task for drawer closing task?

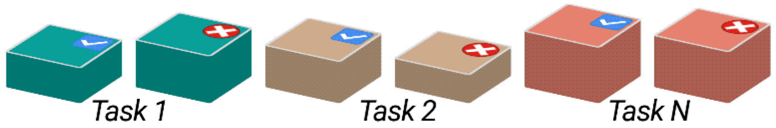
How does that answer change for Q-learning vs Policy Gradient?

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

Example of multi-task Q-learning applied to robotics: MT-Opt



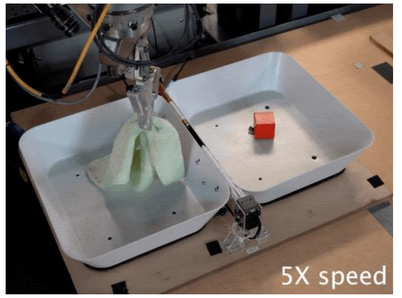




80% avg improvement over baselines across all the ablation tasks (4x improvement over single-task)

~4x avg improvement for tasks with little data

Fine-tunes to a new task (to 92% success) in 1 day



Kalashnikov et al. MT-Opt. CoRL '21

The Plan

Recap & Q-learning

Multi-task imitation and policy gradients

Multi-task Q-learning

Goal-conditioned RL

Goal-conditioned RL with hindsight relabeling

1. Collect data $\mathcal{D}_k = \{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}, \mathbf{s}_g, r_{1:T})\}$ using some policy
2. Store data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k$
3. Perform hindsight relabeling:
   a. Relabel experience in $\mathcal{D}_k$ using last state as goal: $\mathcal{D}'_k = \{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}, \mathbf{s}_T, r'_{1:T}\}$ where $r'_t = -d(\mathbf{s}_t, \mathbf{s}_T)$
   b. Store relabeled data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}'_k$
4. Update policy using replay buffer $\mathcal{D}$

$k++$

g strategies?

e trajectory

Result: exploration challenges alleviated

Kaelbling. Learning to Achieve Goals. IJCAI '93
Andrychowicz et al. Hindsight Experience Replay. NeurIPS '17

Hindsight relabeling for goal-conditioned RL

Example: goal-conditioned RL, simulated robot manipulation

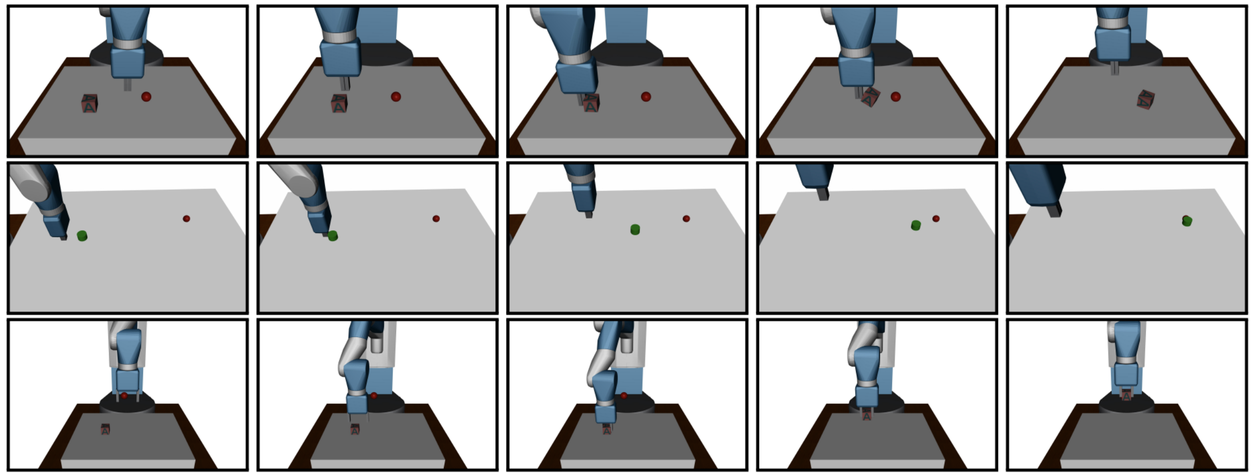
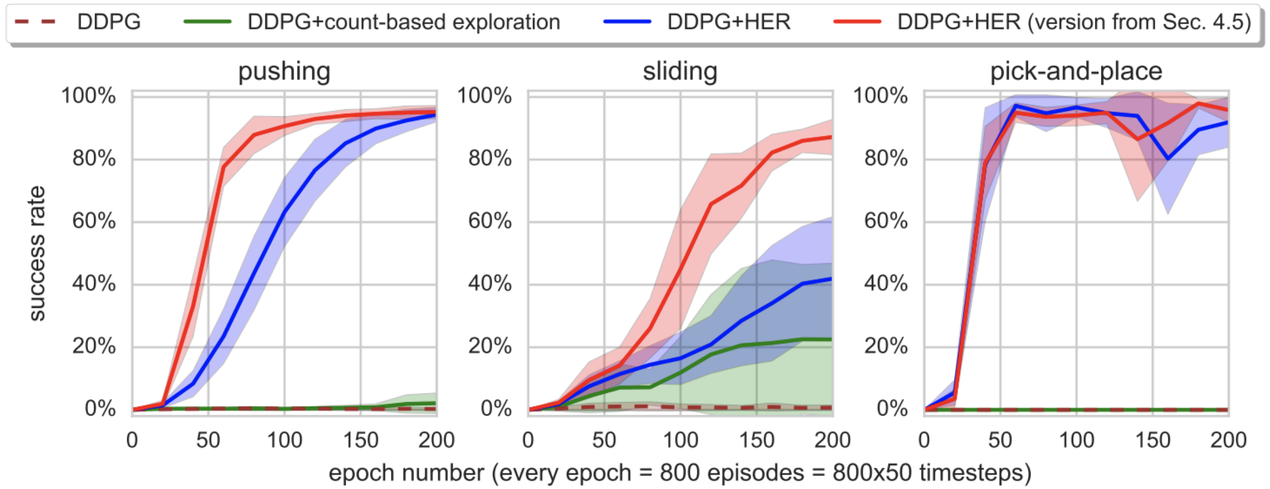


Figure 2: Different tasks: *pushing* (top row), *sliding* (middle row) and *pick-and-place* (bottom row). The red ball denotes the goal position.



Kaelbling. Learning to Achieve Goals. IJCAI '93
Andrychowicz et al. Hindsight Experience Replay. NeurIPS '17

Time Permitting: What about image observations?

Recall: need a distance function between current and goal state!

$$r'_t = -d(\mathbf{s}_t, \mathbf{s}_T)$$

Use binary 0/1 reward? Sparse, but accurate.

Random, **unlabeled interaction** is ***optimal*** under the 0/1 reward of reaching the last state.

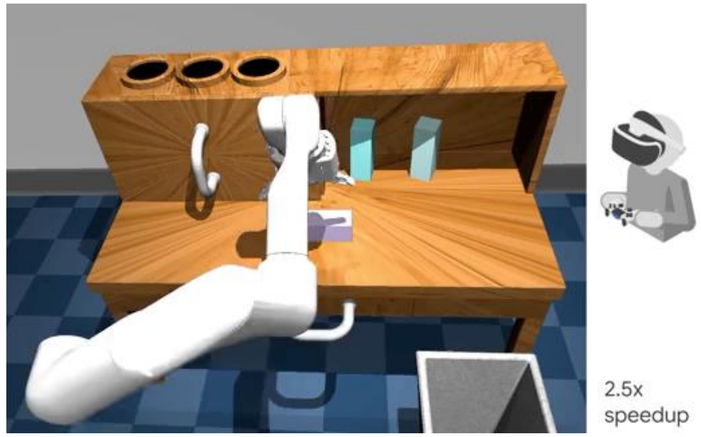
Can we use this insight for **better learning**?

If the data is **optimal**, can we use **supervised imitation learning**?

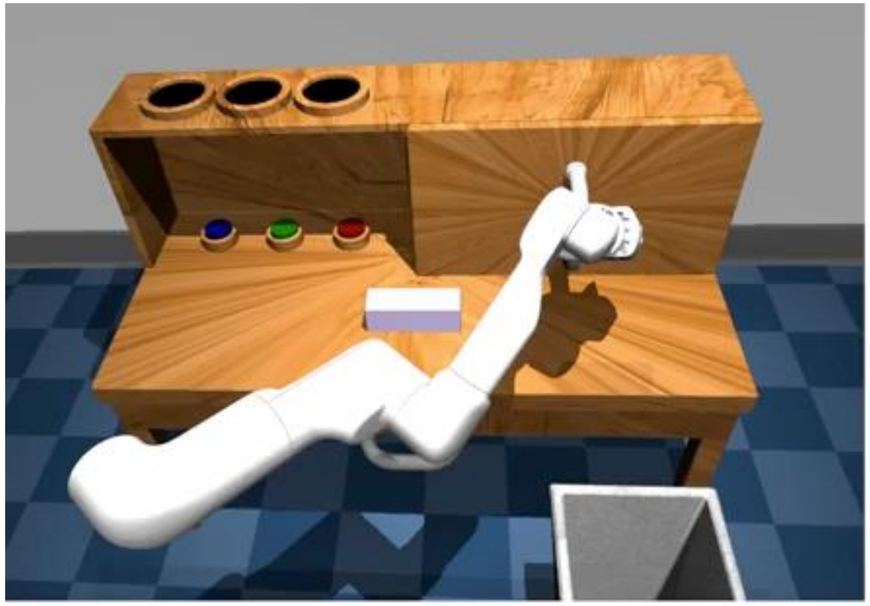
1. Collect data $\mathcal{D}_k = \{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T})\}$ using some policy
2. Perform hindsight relabeling:
   a. Relabel experience in $\mathcal{D}_k$ using last state as goal:
      $\mathcal{D}'_k = \{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}, \mathbf{s}_T, r'_{1:T}\}$ where $r'_t = -d(\mathbf{s}_t, \mathbf{s}_T)$
   b. Store relabeled data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}'_k$
3. Update policy using **supervised imitation on** replay buffer $\mathcal{D}$

Eysenbach, Kumar, Gupta, RL is supervised learning on optimized data. BAIR blogpost, 2020
Ghosh, Gupta et al. Learning to Reach Goals via Iterated Supervised Learning. ICLR '21

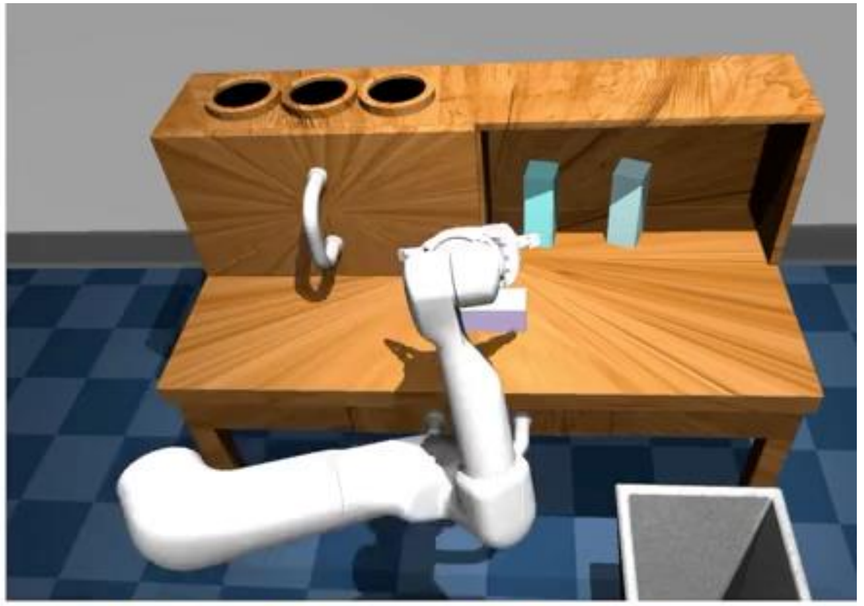
Collect data from "human play", perform goal-conditioned imitation.



2.5x
speedup



Goal



Single Play-LMP policy

Lynch, Khansari, Xiao, Kumar, Tompson, Levine, Sermanet. Learning Latent Plans from Play. '19

The Plan

Recap & Q-learning

Multi-task imitation and policy gradients

Multi-task Q-learning

Goal-conditioned RL

Many Remaining Questions: The Next Three Weeks

How can we use a **model** in multi-task RL?	Model-based RL - **Oct 20**
What about **meta-RL** algorithms?	Meta-RL - **Oct 25**
Can we learn **exploration strategies** across tasks?	Meta-RL: Learning to explore - **Oct 27**
What if **can't collect** data?	Offline RL - **Nov 1**
What about **hierarchies** of tasks?	Hierarchical RL - **Nov 3**

Additional RL Resources

Stanford CS234: Reinforcement Learning

UCL Course from David Silver: Reinforcement Learning

Berkeley CS285: Deep Reinforcement Learning