Exploration: Part 2

CS 285: Deep Reinforcement Learning, Decision Making, and Control

Sergey Levine

Class Notes

1. Homework 4 due today!

Recap: what's the problem?

this is easy (mostly)

Why?

this is impossible

Recap: classes of exploration methods in deep RL

- Optimistic exploration:
  - new state = good state
  - requires estimating state visitation frequencies or novelty
  - typically realized by means of exploration bonuses
- Thompson sampling style algorithms:
  - learn distribution over Q-functions or policies
  - sample and act according to sample
- Information gain style algorithms
  - reason about information gain from visiting new states

Posterior sampling in deep RL

Thompson sampling:

$\theta_1, \ldots, \theta_n \sim \hat{p}(\theta_1, \ldots, \theta_n)$

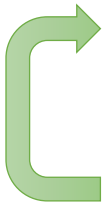
$a = \arg\max_a E_{\theta_a}[r(a)]$

What do we sample?

How do we represent the distribution?

bandit setting: $\hat{p}(\theta_1, \ldots, \theta_n)$ is distribution over *rewards*

MDP analog is the Q-function!



1. sample Q-function Q from $p(Q)$
2. act according to Q for one episode
3. update $p(Q)$

since Q-learning is off-policy, we don't care which Q-function was used to collect data

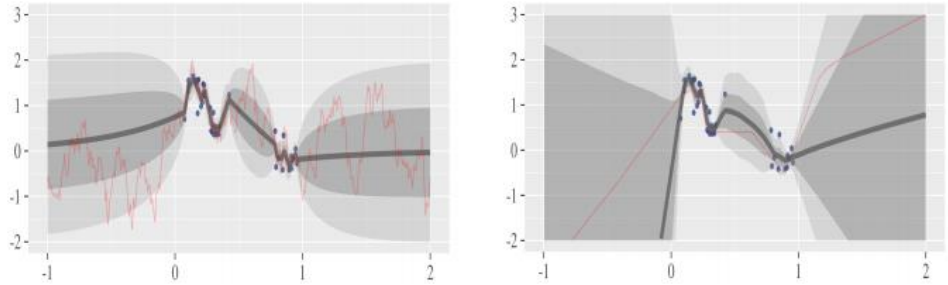
how can we represent a distribution over functions?

Bootstrap

given a dataset $\mathcal{D}$, resample with replacement N times to get $\mathcal{D}_1, \ldots, \mathcal{D}_N$

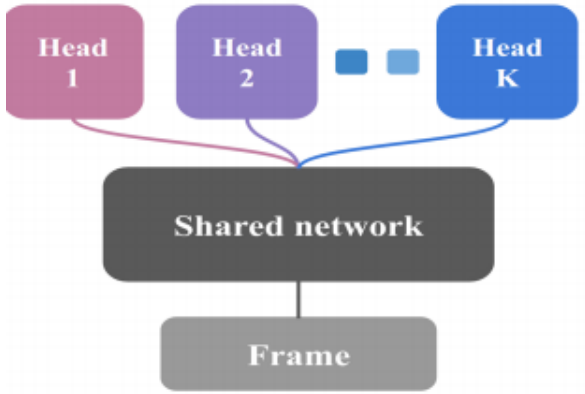
train each model f_{θ_i} on $\mathcal{D}_i$

to sample from $p(\theta)$, sample $i \in [1, \ldots, N]$ and use f_{θ_i}



(b) Gaussian process posterior (c) Bootstrapped neural nets

training N big neural nets is expensive, can we avoid it?

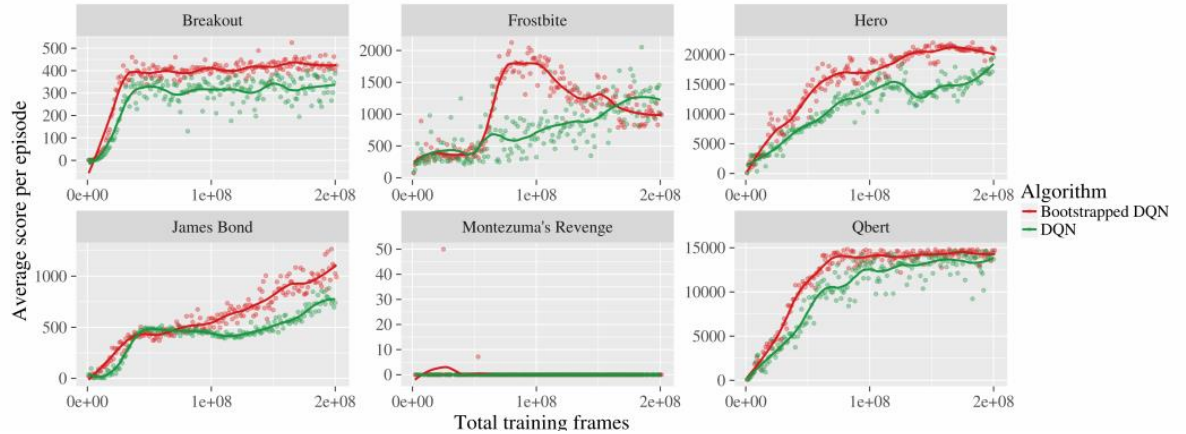


Osband et al. “Deep Exploration via Bootstrapped DQN”

Why does this work?

Exploring with random actions (e.g., epsilon-greedy): oscillate back and forth, might not go to a coherent or interesting place

Exploring with random Q-functions: commit to a randomized but internally consistent strategy for an entire episode



+ no change to original reward function

- very good bonuses often do better

Osband et al. “Deep Exploration via Bootstrapped DQN”

Reasoning about information gain (approximately)

Info gain: $\mathrm{IG}(z, y|a)$

information gain about *what*?	
information gain about reward $r(\mathbf{s}, \mathbf{a})$?	not very useful if reward is sparse
state density $p(\mathbf{s})$?	a bit strange, but somewhat makes sense!
information gain about dynamics $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$?	good proxy for *learning* the MDP, though still heuristic

Generally intractable to use exactly, regardless of what is being estimated!

Reasoning about information gain (approximately)

Generally intractable to use exactly, regardless of what is being estimated

A few approximations:

prediction gain: $\log p_{\theta'}(\mathbf{s}) - \log p_{\theta}(\mathbf{s})$ (Schmidhuber '91, Bellemare '16)

intuition: if density changed a lot, the state was novel

variational inference: (Houthooft et al. "VIME")

IG can be equivalently written as $D_{\mathrm{KL}}(p(z|y)\|p(z))$

learn about *transitions* $p_\theta(s_{t+1}|s_t, a_t)$: $z = \theta$

$y = (s_t, a_t, s_{t+1})$

$$D_{\mathrm{KL}}(p(\theta|h, s_t, a_t, s_{t+1})\|p(\theta|h))$$

model parameters for $p_\theta(s_{t+1}|s_t, a_t)$

history of all prior transitions

newly observed transition

intuition: a transition is more informative if it causes belief over θ to change

idea: use variational inference to estimate $q(\theta|\phi) \approx p(\theta|h)$

given new transition (s, a, s'), update ϕ to get ϕ'

Reasoning about information gain (approximately)

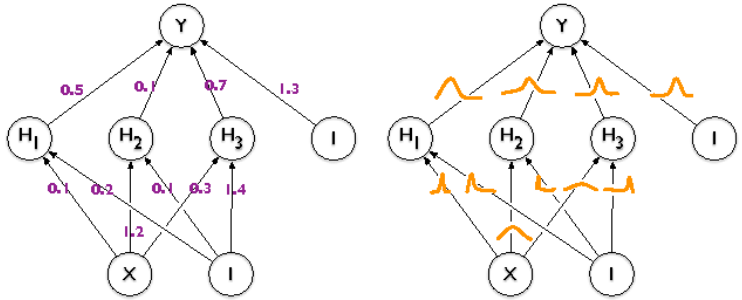
VIME implementation:

IG can be equivalently written as $D_{\mathrm{KL}}(p(\theta|h, \underline{s_t, a_t, s_{t+1}})\|p(\theta|h))$

model parameters for $p_\theta(s_{t+1}|s_t, a_t)$

history of all prior transitions

newly observed transition



$q(\theta|\phi) \approx p(\theta|h)$ specifically, optimize variational lower bound $D_{\mathrm{KL}}(q(\theta|\phi)\|p(h|\theta)p(\theta))$

represent $q(\theta|\phi)$ as product of independent Gaussian parameter distributions

with mean ϕ (see Blundell et al. "Weight uncertainty in neural networks")

given new transition (s, a, s'), update ϕ to get ϕ'

this corresponds to updating the network weight means and variances

use $D_{\mathrm{KL}}(q(\theta|\phi')\|q(\theta|\phi))$ as approximate bonus

$$p(\theta|\mathcal{D}) = \prod_i p(\theta_i|\mathcal{D})$$

$$p(\theta_i|\mathcal{D}) = \mathcal{N}(\mu_i, \sigma_i)$$

ϕ

Houthooft et al. "VIME"

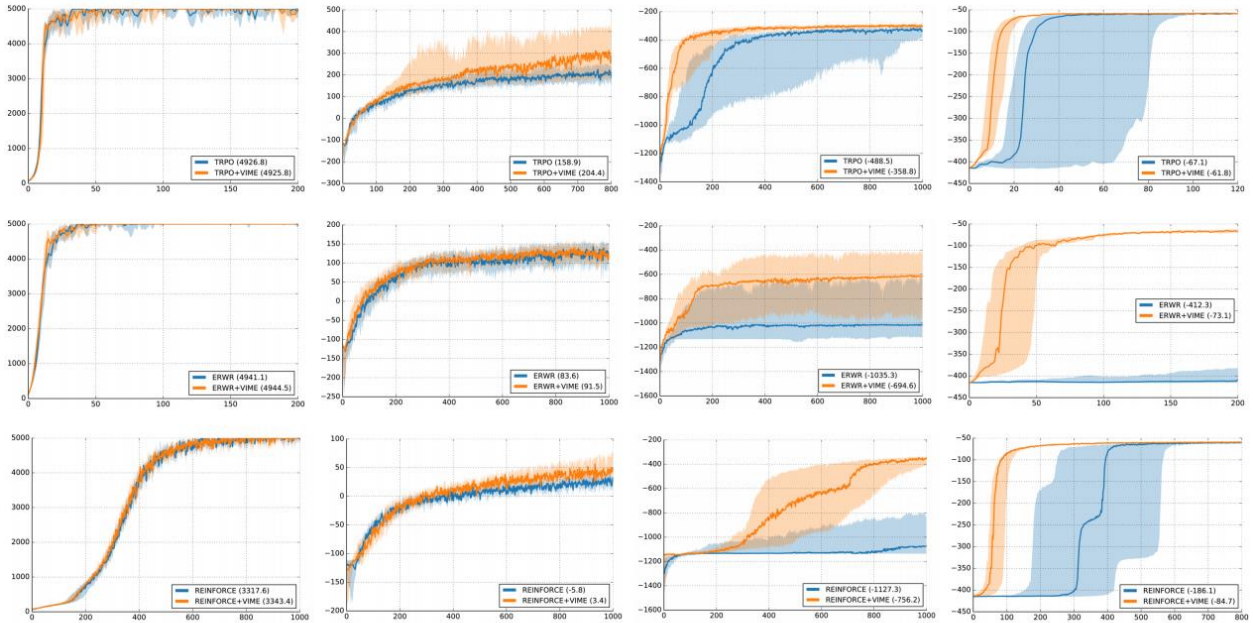
Reasoning about information gain (approximately)

VIME implementation:

IG can be equivalently written as $D_{\mathrm{KL}}(p(\theta|h, s_t, a_t, s_{t+1})\|p(\theta|h))$

$q(\theta|\phi) \approx p(\theta|h)$ specifically, optimize variational lower bound $D_{\mathrm{KL}}(q(\theta|\phi)\|p(h|\theta)p(\theta))$

use $D_{\mathrm{KL}}(q(\theta|\phi')\|q(\theta|\phi))$ as approximate bonus



(a) CartPole (b) CartPoleSwingup (c) DoublePendulum (d) MountainCar

Approximate IG:

+ appealing mathematical formalism

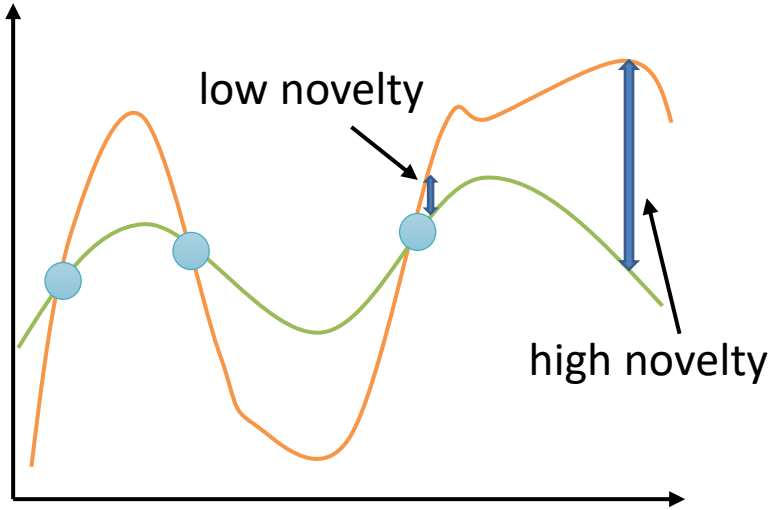
- models are more complex, generally harder to use effectively

Houthooft et al. “VIME”

Exploration with model errors

$D_{\mathrm{KL}}(q(\theta|\phi')\|q(\theta|\phi))$ can be seen as change in network (mean) parameters ϕ

if we forget about IG, there are many other ways to measure this



Stadie et al. 2015:

- encode image observations using auto-encoder
- build predictive model on auto-encoder latent states
- use model error as exploration bonus

Schmidhuber et al. (see, e.g. “Formal Theory of Creativity, Fun, and Intrinsic Motivation):

- exploration bonus for model error
- exploration bonus for model gradient
- many other variations

Many others!

Recap: classes of exploration methods in deep RL

- Optimistic exploration:
  - Exploration with counts and pseudo-counts
  - Different models for estimating densities
- Thompson sampling style algorithms:
  - Maintain a distribution over models via bootstrapping
  - Distribution over Q-functions
- Information gain style algorithms
  - Generally intractable
  - Can use variational approximation to information gain

Suggested readings

Schmidhuber. (1992). **A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers.**

Stadie, Levine, Abbeel (2015). **Incentivizing Exploration in Reinforcement Learning with Deep Predictive Models.**

Osband, Blundell, Pritzel, Van Roy. (2016). **Deep Exploration via Bootstrapped DQN.**

Houthooft, Chen, Duan, Schulman, De Turck, Abbeel. (2016). **VIME: Variational Information Maximizing Exploration.**

Bellemare, Srinivasan, Ostroviski, Schaul, Saxton, Munos. (2016). **Unifying Count-Based Exploration and Intrinsic Motivation.**

Tang, Houthooft, Foote, Stooke, Chen, Duan, Schulman, De Turck, Abbeel. (2016). **#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning.**

Fu, Co-Reyes, Levine. (2017). **EX2: Exploration with Exemplar Models for Deep Reinforcement Learning.**

Break

Imitation vs. Reinforcement Learning

imitation learning

- Requires demonstrations
- Must address distributional shift
- Simple, stable supervised learning
- Only as good as the demo

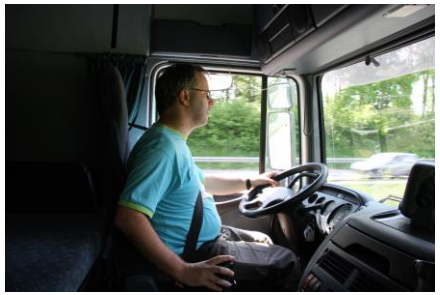
reinforcement learning

- Requires reward function
- Must address exploration
- Potentially non-convergent RL
- Can become arbitrarily good

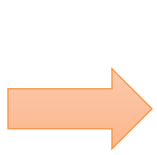
Can we get the best of both?

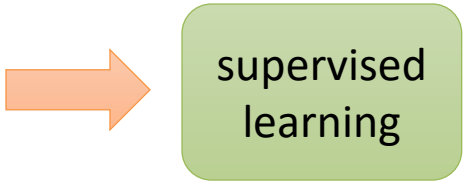
e.g., what if we have demonstrations *and* rewards?

Imitation Learning

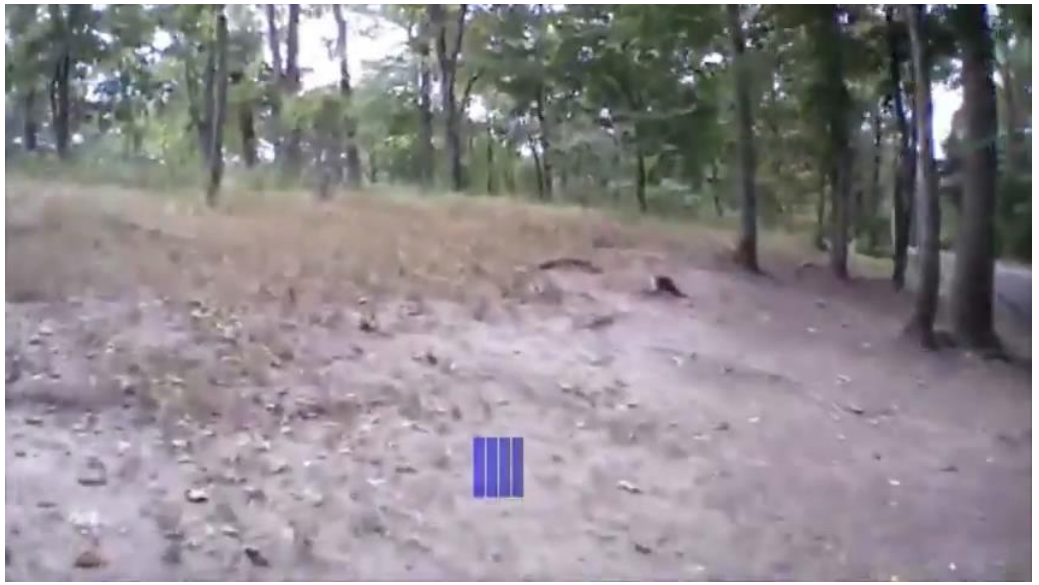


$\mathbf{o}_t$
$\mathbf{a}_t$

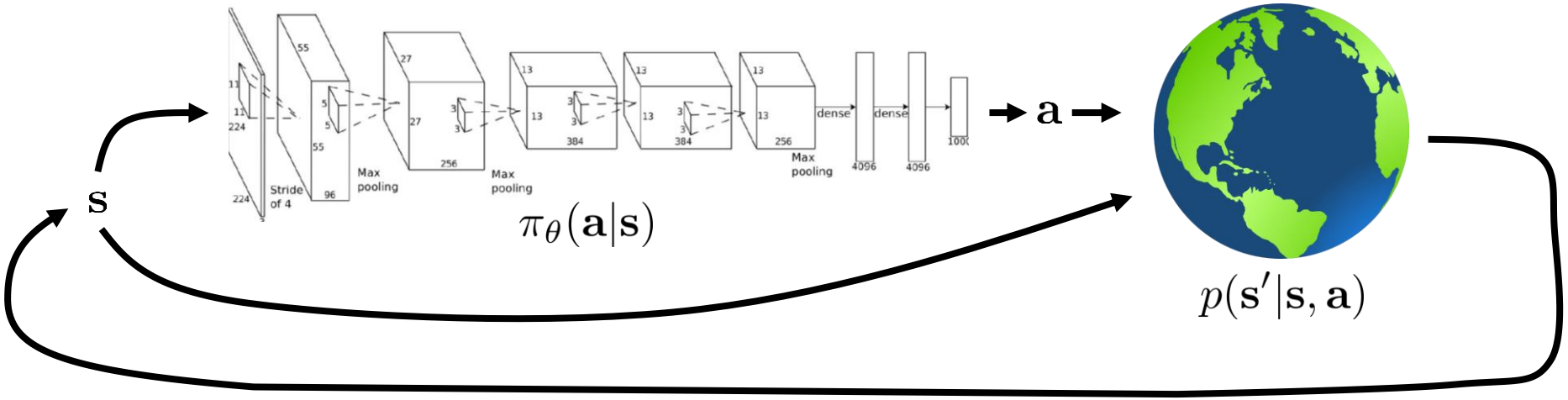




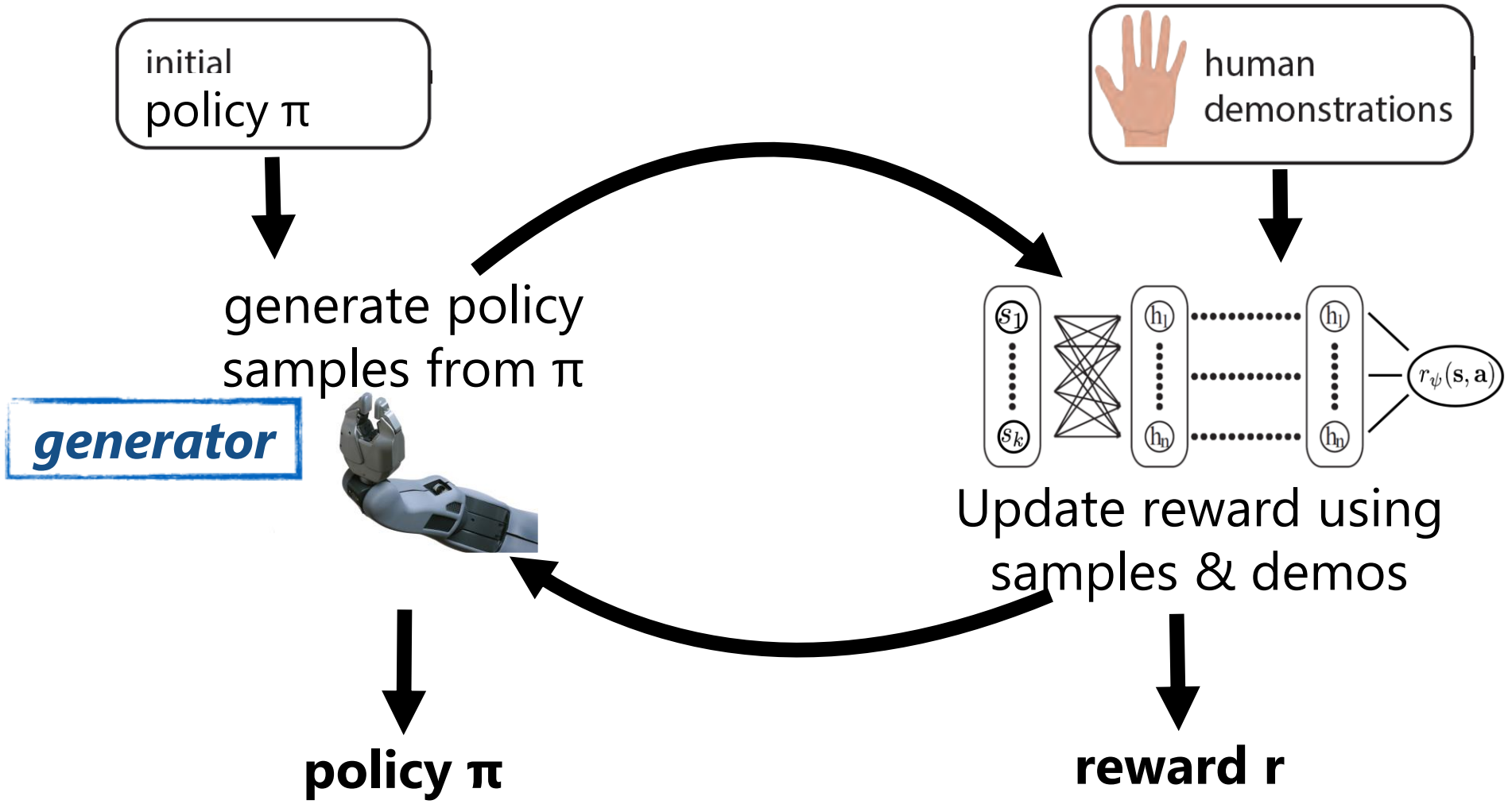
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$



Reinforcement Learning

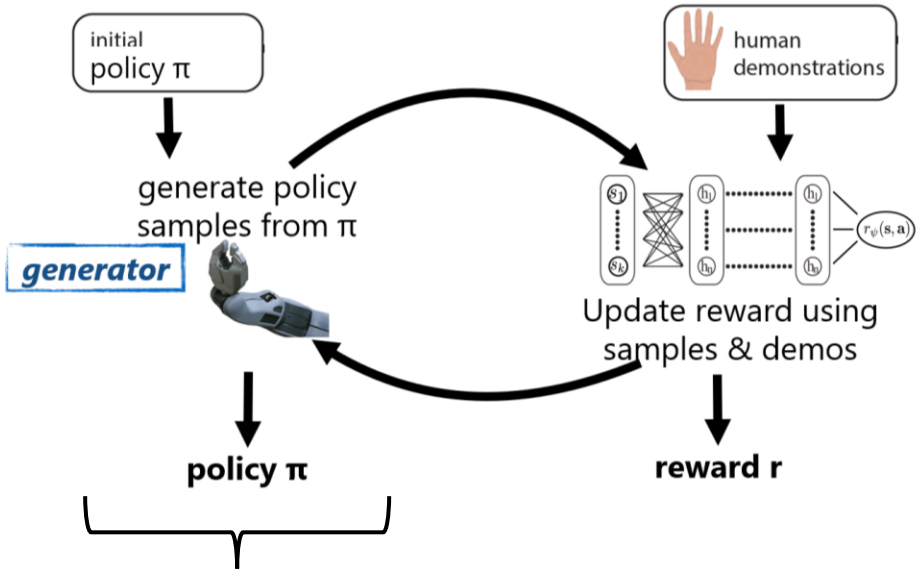


Addressing distributional shift with RL?



Addressing distributional shift with RL?

IRL *already* addresses distributional shift via RL



this part is regular “forward” RL

But it doesn’t use a known reward function!

Simplest combination: pretrain & finetune

- Demonstrations can overcome exploration: show us how to do the task
- Reinforcement learning can improve beyond performance of the demonstrator
- Idea: initialize with imitation learning, then finetune with reinforcement learning!

1. collected demonstration data $(\mathbf{s}_i, \mathbf{a}_i)$
2. initialize π_θ as $\max_\theta \sum_i \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i)$
3. run π_θ to collect experience
4. improve π_θ with any RL algorithm

Simplest combination: pretrain & finetune

Muelling et al. ‘13

Simplest combination: pretrain & finetune

Pretrain & finetune

1. collected demonstration data $(\mathbf{s}_i, \mathbf{a}_i)$
2. initialize π_θ as $\max_\theta \sum_i \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i)$
3. run π_θ to collect experience
4. improve π_θ with any RL algorithm

vs. DAgger

1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \ldots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

What's the problem?

Pretrain & finetune

1. collected demonstration data $(\mathbf{s}_i, \mathbf{a}_i)$
2. initialize π_θ as $\max_\theta \sum_i \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i)$
3. run π_θ to collect experience ← can be very bad (due to distribution shift)
4. improve π_θ with any RL algorithm ← first batch of (very) bad data can destroy initialization

Can we avoid forgetting the demonstrations?

Off-policy reinforcement learning

- Off-policy RL can use any data
- If we let it use demonstrations as off-policy samples, can that mitigate the exploration challenges?
  - Since demonstrations are provided as data in every iteration, they are never forgotten
  - But the policy can still become *better* than the demos, since it is not forced to mimic them

off-policy policy gradient (with importance sampling)

off-policy Q-learning

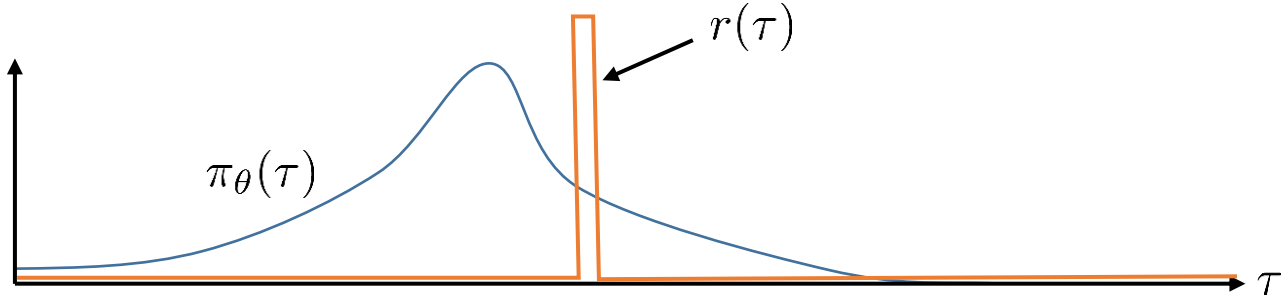
Policy gradient with demonstrations

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

includes demonstrations *and* experience

Why is this a good idea? Don't we want on-policy samples?

best sampling distribution should have high reward!



optimal importance sampling

say we want $E_{p(x)}[f(x)]$

$E_{p(x)}[f(x)] \approx \frac{1}{N} \sum_i \frac{p(x_i)}{q(x_i)} f(x_i)$

which $q(x)$ gives lowest variance?

answer: $q(x) \propto p(x)|f(x)|$

Policy gradient with demonstrations

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

How do we construct the sampling distribution?

problem 1: which distribution did the demonstrations come from?

option 1: use supervised behavior cloning to approximate π_{demo}

option 2: assume Diract delta: $\pi_{\text{demo}}(\tau) = \frac{1}{N}\delta(\tau \in \mathcal{D})$

this works best with self-normalized importance sampling

standard IS

$$E_{p(x)}[f(x)] \approx \frac{1}{N} \sum_i \frac{p(x_i)}{q(x_i)} f(x_i)$$

self-normalized IS

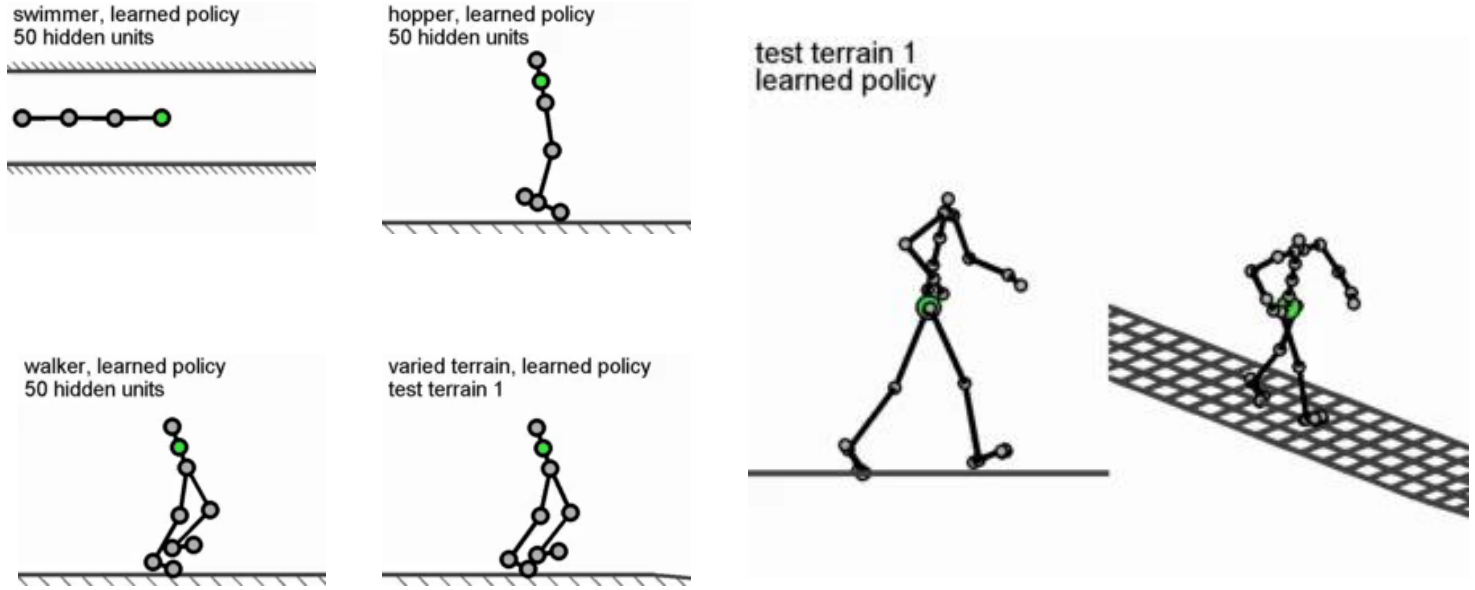
$$E_{p(x)}[f(x)] \approx \frac{1}{\sum_j \frac{p(x_j)}{q(x_j)}} \sum_i \frac{p(x_i)}{q(x_i)} f(x_i)$$

problem 2: what to do if we have multiple distributions?

fusion distribution: $q(x) = \frac{1}{M} \sum_i q_i(x)$

Example: importance sampling with demos

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



Levine, Koltun ’13. “Guided policy search”

Q-learning with demonstrations

- Q-learning is *already* off-policy, no need to bother with importance weights!
- Simple solution: drop demonstrations into the replay buffer

full Q-learning with replay buffer:

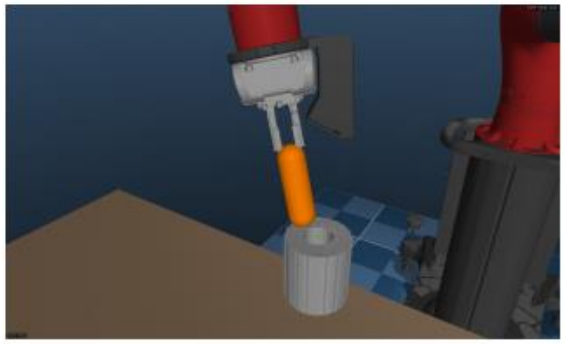
initialize $\mathcal{B}$ to contain the demonstration data

1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy, add it to $\mathcal{B}$

$K\times$

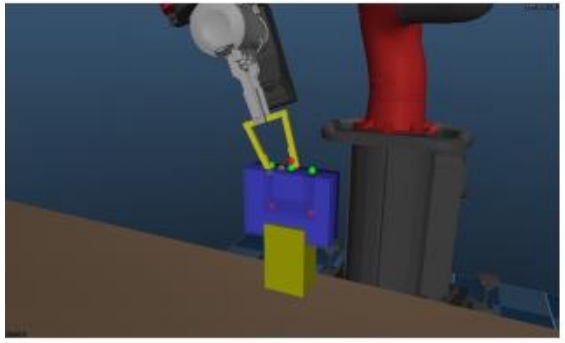
2. sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ from $\mathcal{B}$
3. $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)])$

Q-learning with demonstrations

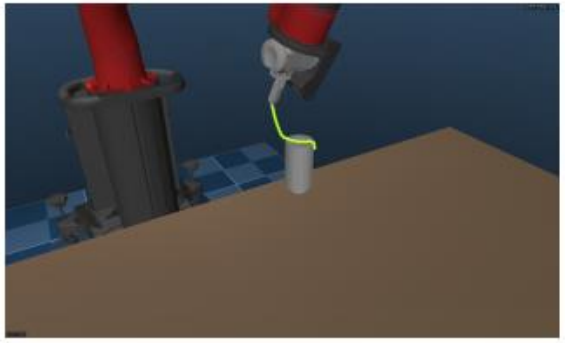


(a) Peg Insertion Task.

(b) Hard-drive Task.



(c) Clip Insertion Task



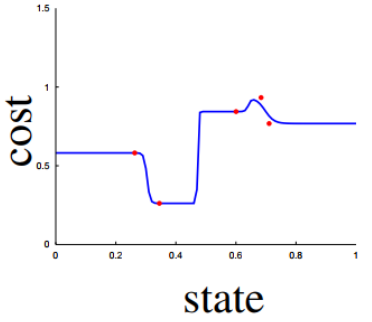
(d) Cable Insertion Task.

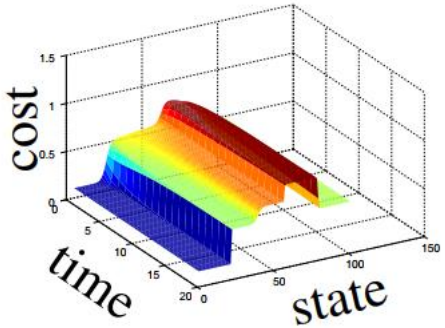
Vecerik et al., ‘17, “Leveraging Demonstrations for Deep Reinforcement Learning…”

What's the problem?

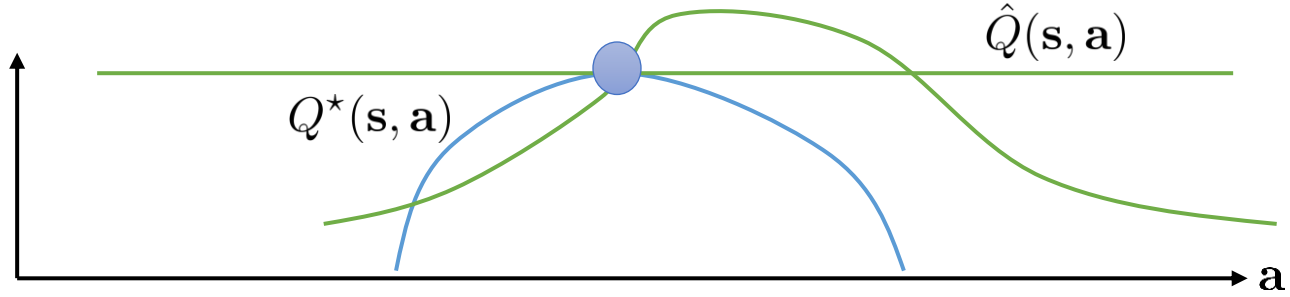
Importance sampling: recipe for getting stuck

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





Q-learning: just good data is not enough

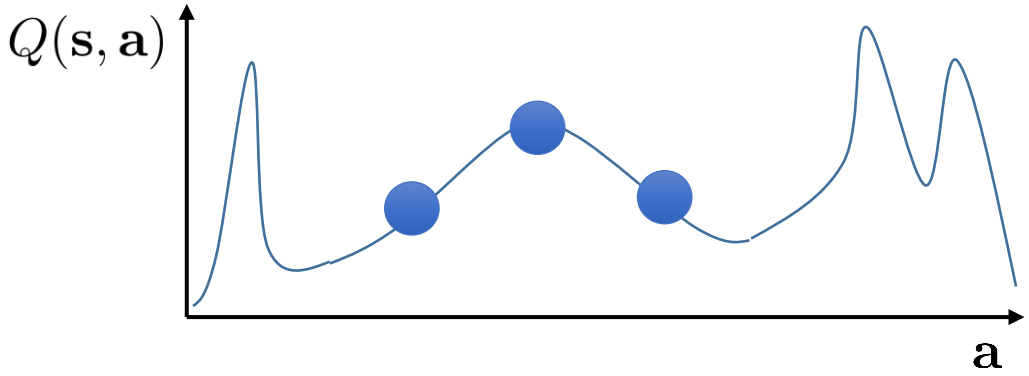


More problems with Q learning

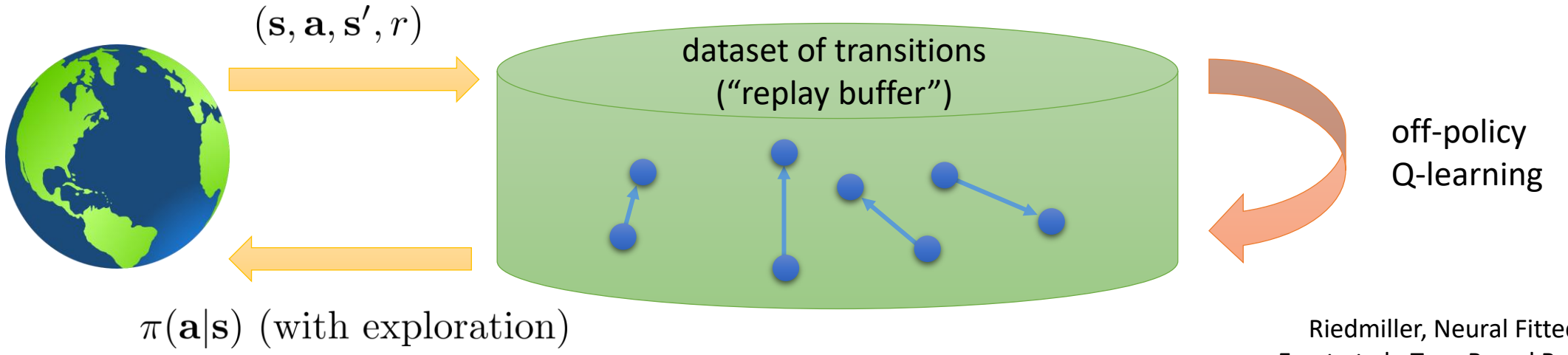
$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}')$$

what action will this pick?

$Q(\mathbf{s}, \mathbf{a})$ is trained on $(\mathbf{s}, \mathbf{a}) \sim \beta(\mathbf{s}, \mathbf{a})$



if $\mathbf{a}^\star = \arg\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$ makes $\beta(\mathbf{s}, \mathbf{a}^\star)$ small
we end up training on garbage!



See, e.g.
Riedmiller, Neural Fitted Q-Iteration '05
Ernst et al., Tree-Based Batch Mode RL '05

More problems with Q learning

~~$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}')$~~

$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + E_{\mathbf{a}'\sim\pi_{\text{new}}}[Q(\mathbf{s}',\mathbf{a}')]$

how to pick $\pi_{\text{new}}(\mathbf{a}|\mathbf{s})$?

option 1: stay *close* to β

e.g. $D_{\text{KL}}(\pi_{\text{new}}(\cdot|\mathbf{s})\|\beta(\cdot|\mathbf{s})) \leq \epsilon$

issue 1: we don't know β

issue 2: this is *way* too conservative

key idea: constrain to *support* of β

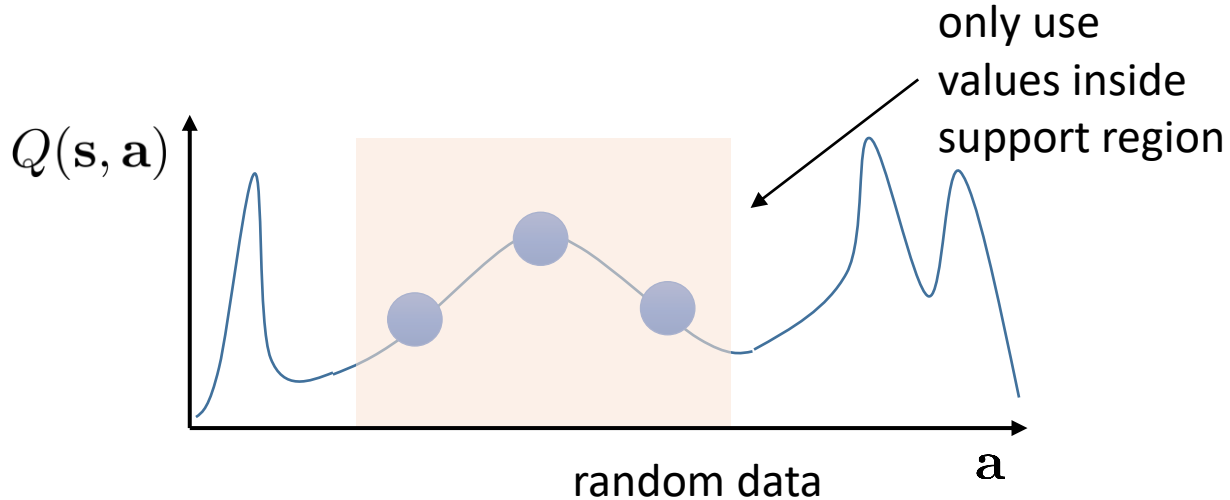
$$\max_{\pi\in\Delta_{|S|}} \mathbb{E}_{a\sim\pi(\cdot|s)}[\hat{Q}_k(s,a)] - \lambda\sqrt{\text{var}_k\, \mathbb{E}_{a\sim\pi(\cdot|s)}[\hat{Q}_k(s,a)]}$$

pessimistic w.r.t. epistemic uncertainty

$$\text{s.t. } \mathbb{E}_{s\sim\mathcal{D}}[\text{MMD}(\mathcal{D}(s),\pi(\cdot|s))] \leq \varepsilon$$

support constraint

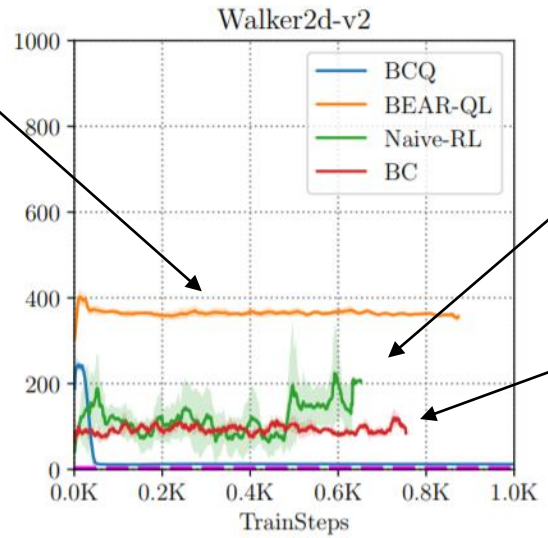
$Q(\mathbf{s},\mathbf{a})$ is trained on $(\mathbf{s},\mathbf{a}) \sim \beta(\mathbf{s},\mathbf{a})$



BEAR

naïve RL

distrib. matching (BCQ)



See: Kumar, Fu, Tucker, Levine. **Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction.**

See also: Fujimoto, Meger, Precup**. Off-Policy Deep Reinforcement Learning without Exploration.**

So far…

- Pure imitation learning
  - Easy and stable supervised learning
  - Distributional shift
  - No chance to get better than the demonstrations
- Pure reinforcement learning
  - Unbiased reinforcement learning, can get arbitrarily good
  - Challenging exploration and optimization problem
- Initialize & finetune
  - Almost the best of both worlds
  - …but can forget demo initialization due to distributional shift
- Pure reinforcement learning, with demos as off-policy data
  - Unbiased reinforcement learning, can get arbitrarily good
  - Demonstrations don’t always help
- Can we strike a compromise? A little bit of supervised, a little bit of RL?

Imitation as an auxiliary loss function

imitation objective: $\sum_{(\mathbf{s},\mathbf{a})\in\mathcal{D}_{\text{demo}}} \log \pi_\theta(\mathbf{a}|\mathbf{s})$ (or some variant of this)

RL objective: $E_{\pi_\theta}[r(\mathbf{s},\mathbf{a})]$ (or some variant of this)

hybrid objective: $E_{\pi_\theta}[r(\mathbf{s},\mathbf{a})] + \lambda \sum_{(\mathbf{s},\mathbf{a})\in\mathcal{D}_{\text{demo}}} \log \pi_\theta(\mathbf{a}|\mathbf{s})$

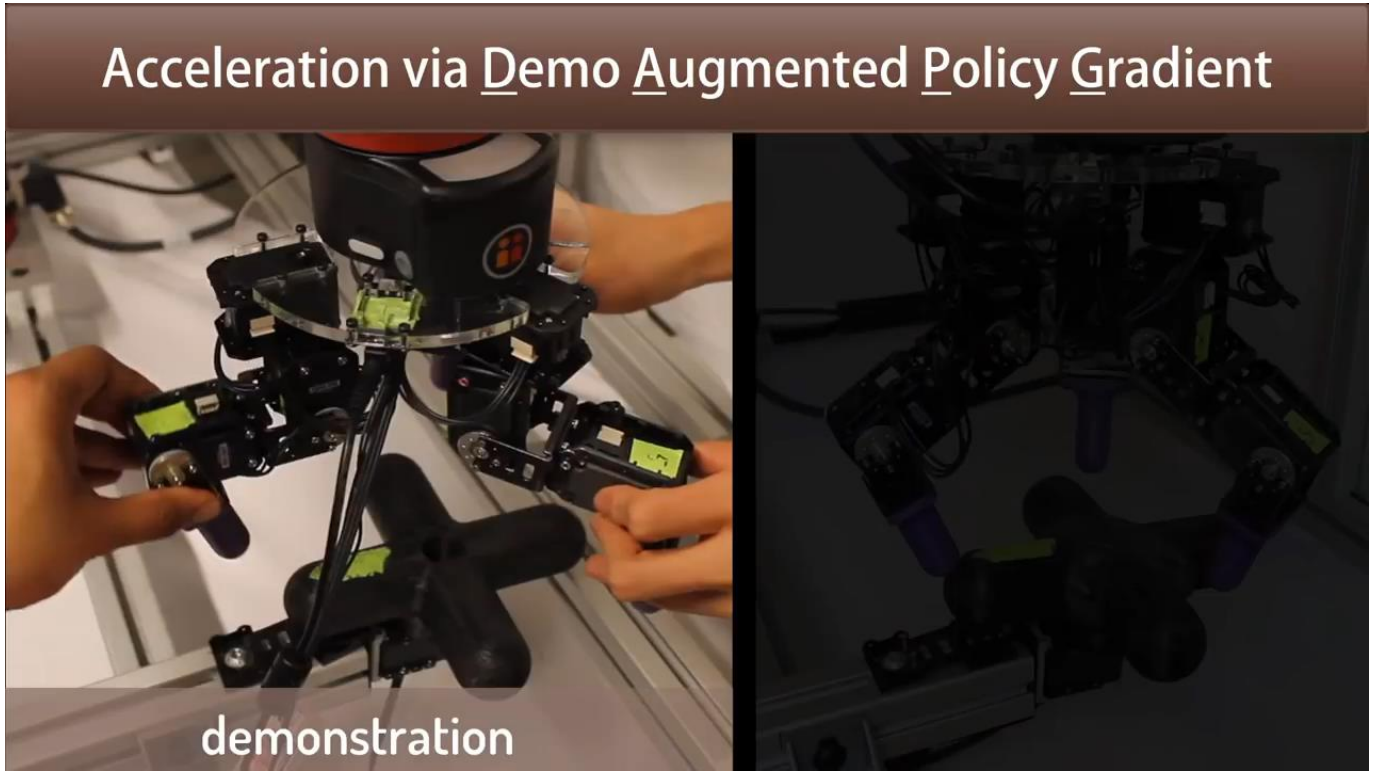
need to be careful in choosing this weight

Example: hybrid policy gradient

standard policy gradient

$$g_{aug} = \sum_{(s,a)\in\rho_\pi} \nabla_\theta \ln \pi_\theta(a|s) A^\pi(s,a) + \sum_{(s,a^*)\in\rho_D} \nabla_\theta \ln \pi_\theta(a^*|s) w(s,a^*)$$

increase demo likelihood



Rajeswaran et al., ‘17, “Learning Complex Dexterous Manipulation…”

Example: hybrid Q-learning

$$J(Q) = J_{DQ}(Q) + \lambda_1 J_n(Q) + \lambda_2 J_E(Q) + \lambda_3 J_{L2}(Q).$$

regularization loss

because why not…

Q-learning loss

n-step Q-learning loss

$$J_E(Q) = \max_{a \in A}[Q(s,a) + l(a_E, a)] - Q(s, a_E)$$

margin-based loss on example action

Hester et al., ‘17, “Learning from Demonstrations…”

What's the problem?

hybrid objective: $E_{\pi_\theta}[r(\mathbf{s}, \mathbf{a})] + \lambda \sum_{(\mathbf{s},\mathbf{a}) \in \mathcal{D}_{\text{demo}}} \log \pi_\theta(\mathbf{a}|\mathbf{s})$

- Need to tune the weight
- The design of the objective, esp. for imitation, takes a lot of care
- Algorithm becomes problem-dependent

- Pure imitation learning
  - Easy and stable supervised learning
  - Distributional shift
  - No chance to get better than the demonstrations
- Pure reinforcement learning
  - Unbiased reinforcement learning, can get arbitrarily good
  - Challenging exploration and optimization problem
- Initialize & finetune
  - Almost the best of both worlds
  - …but can forget demo initialization due to distributional shift
- Pure reinforcement learning, with demos as off-policy data
  - Unbiased reinforcement learning, can get arbitrarily good
  - Demonstrations don’t always help
- Hybrid objective, imitation as an “auxiliary loss”
  - Like initialization & finetuning, almost the best of both worlds
  - No forgetting
  - But no longer pure RL, may be biased, may require lots of tuning