Deep Reinforcement Learning
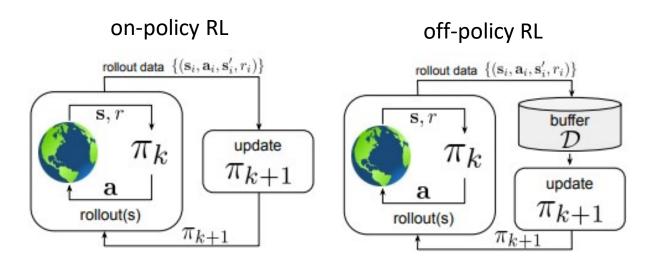
17: Offline Reinforcement Learning Part 2

Eric Benhamou David Saltiel

Acknowledgement

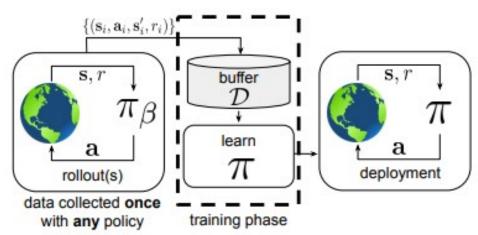These materials are based on the seminal course of Sergey Levine CS285

Offline Reinforcement Learning

on-policy RL

off-policy RL

offline reinforcement learning

Formally:

$$\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$$

$$\mathbf{s} \sim d^{\pi_\beta}(\mathbf{s})$$
$$\mathbf{a} \sim \pi_\beta(\mathbf{a}|\mathbf{s}) \quad \longleftarrow \quad \text{generally \textbf{not} known}$$
$$\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$$
$$r \leftarrow r(\mathbf{s}, \mathbf{a})$$

RL objective: $\displaystyle \max_\pi \sum_{t=0}^{T} E_{\mathbf{s}_t \sim d^\pi(\mathbf{s}), \mathbf{a}_t \sim \pi(\mathbf{a}|\mathbf{s})}[\gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$

Where do we suffer from distribution shift?

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

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

what is the objective?

$$\min_{Q} E_{(\mathbf{s}, \mathbf{a}) \sim \pi_\beta(\mathbf{s}, \mathbf{a})} \left[(Q(\mathbf{s}, \mathbf{a}) - y(\mathbf{s}, \mathbf{a}))^2 \right]$$

behavior policy

target value

expect good accuracy when $\pi_\beta(\mathbf{a}|\mathbf{s}) = \pi_{\mathrm{new}}(\mathbf{a}|\mathbf{s})$
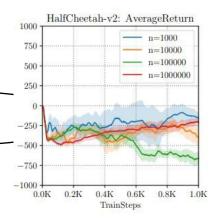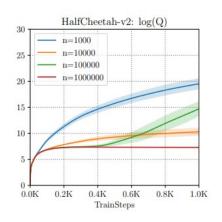
how often does *that* happen?

even *worse*: $\pi_{\mathrm{new}} = \arg\max_\pi E_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})]$

(what if we pick $\mathbf{x}^\star \leftarrow \arg\max_\mathbf{x} f_\theta(\mathbf{x})$?)

how well it does

how well it *thinks* it does (Q-values)

3

Kumar, Fu, Tucker, Levine. **Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction.** NeurIPS '19

How do prior methods address this?

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

$$\pi_{\mathrm{new}}(\mathbf{a}|\mathbf{s}) = \arg\max_{\pi} E_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})] \ \mathrm{s.t.} \ D_{\mathrm{KL}}(\pi\|\pi_{\beta}) \leq \epsilon$$

This solves distribution shift, right?

No more erroneous values?

Issue 1: we usually don't know the behavior policy $\pi_{\beta}(\mathbf{a}|\mathbf{s})$

- human-provided data
- data from hand-designed controller
- data from many past RL runs
- all of the above

Issue 2: this is both *too pessimistic* and *not pessimistic enough*

"policy constraint" method

very old idea (but it had no single name?)

Todorov et al. [passive dynamics in linearly-solvable MDPs]

Kappen et al. [KL-divergence control, etc.]

trust regions, covariant policy gradients, natural policy gradients, etc.

used in some form in recent papers:

Fox et al. '15 ("Taming the Noise...")

Fujimoto et al. '18 ("Off Policy...")

Jaques et al. '19 ("Way Off Policy...")

Kumar et al. '19 ("Stabilizing...")
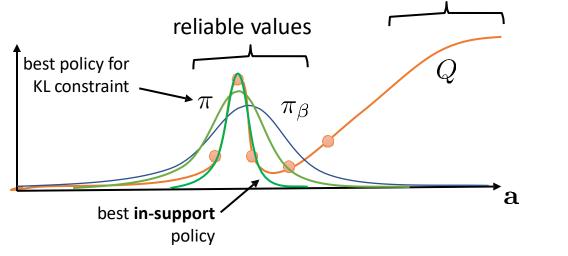
Wu et al. '19 ("Behavior Regularized...")

Levine, Kumar, Tucker, Fu. **Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems.** '20

Explicit policy constraint methods

What kinds of constraints can we use?

KL-divergence: $D_{\mathrm{KL}}(\pi \| \pi_\beta)$

+ easy to implement (more on this later)

- not necessarily what we want

support constraint: $\pi(\mathbf{a}|\mathbf{s}) \geq 0$ only if $\pi_\beta(\mathbf{a}|\mathbf{s}) \geq \epsilon$

can approximate with MMD

- significantly more complex to implement

+ much closer to what we really want

unreliable OOD values

reliable values

best policy for KL constraint π π_β

Q

best **in-support** policy

$\mathbf{a}$

For more information, see:

Levine, Kumar, Tucker, Fu. **Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems.** '20

Kumar, Fu, Tucker, Levine. **Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction.** '19

Wu, Tucker, Nachum. **Behavior Regularized Offline Reinforcement Learning.** `19

Explicit policy constraint methods

How do we implement constraints?

Lagrange multiplier

easy to compute and differentiate
for Gaussian or categorical policies

1. Modify the actor objective

$$\theta \leftarrow \arg\max_{\theta} E_{\mathbf{s} \sim D}\left[E_{\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})]\right]$$

$$\theta \leftarrow \arg\max_{\theta} E_{\mathbf{s} \sim D}\left[E_{\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a}) + \lambda \log \pi_\beta(\mathbf{a}|\mathbf{s})] + \lambda \mathcal{H}(\pi(\mathbf{a}|\mathbf{s}))\right]$$

$$D_{\mathrm{KL}}(\pi \| \pi_\beta) = E_\pi[\log \pi(\mathbf{a}|\mathbf{s}) - \log \pi_\beta(\mathbf{a}|\mathbf{s})] = -E_\pi[\log \pi_\beta(\mathbf{a}|\mathbf{s})] - \mathcal{H}(\pi)$$

2. Modify the reward function

$$\bar{r}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) - D(\pi, \pi_\beta)$$

simple modification to directly penalize divergence
also accounts for **future** divergence

See: Wu, Tucker, Nachum. **Behavior Regularized Offline Reinforcement Learning.** `19

generally, the best modern offline RL methods do not do either of these things

Implicit policy constraint methods

$$\pi_{\text{new}}(\mathbf{a}|\mathbf{s}) = \arg\max_{\pi} E_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] \text{ s.t. } D_{\text{KL}}(\pi\|\pi_\beta) \leq \epsilon$$

See also:
Peters et al. (REPS)
Rawlik et al. ("psi-learning")
...many follow-ups

$$\pi^\star(\mathbf{a}|\mathbf{s}) = \frac{1}{Z(\mathbf{s})}\pi_\beta(\mathbf{a}|\mathbf{s})\exp\left(\frac{1}{\lambda}A^\pi(\mathbf{s},\mathbf{a})\right)$$

straightforward to
show via duality

approximate via **weighted** max likelihood!

$$\pi_{\text{new}}(\mathbf{a}|\mathbf{s}) = \arg\max_{\pi} E_{(\mathbf{s},\mathbf{a})\sim\pi_\beta}\left[\log\pi(\mathbf{a}|\mathbf{s})\overbrace{\frac{1}{Z(\mathbf{s})}\exp\left(\frac{1}{\lambda}A^{\pi_{\text{old}}}(\mathbf{s},\mathbf{a})\right)}^{w(\mathbf{s},\mathbf{a})}\right]$$

samples from dataset
$\mathbf{a}\sim\pi_\beta(\mathbf{a}|\mathbf{s})$

critic can be used
to give us this

Peng*, Kumar*, Levine. **Advantage-Weighted Regression.** '19

Nair, Dalal, Gupta, Levine. **Accelerating Online Reinforcement Learning with Offline Datasets.** '20

Implicit policy constraint methods

$$\mathcal{L}_C(\phi) = E_{(\mathbf{s},\mathbf{a},\mathbf{s}') \sim D} \left[\left(Q_\phi(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + \gamma E_{\mathbf{a}' \sim \pi_\theta(\mathbf{a}'|\mathbf{s}')}[Q_\phi(\mathbf{s}',\mathbf{a}')])) \right)^2 \right]$$

$$\mathcal{L}_A(\theta) = -E_{(\mathbf{s},\mathbf{a}) \sim \pi_\beta} \left[\log \pi_\theta(\mathbf{a}|\mathbf{s}) \frac{1}{Z(\mathbf{s})} \exp\left(\frac{1}{\lambda} A^{\pi_{\mathrm{old}}}(\mathbf{s},\mathbf{a}) \right) \right]$$

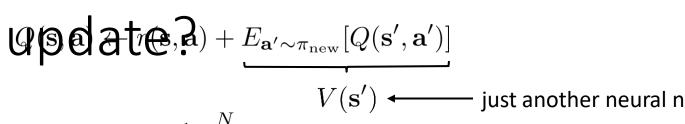1. $\phi \leftarrow \phi - \alpha \nabla_\phi \mathcal{L}_C(\phi)$

2. $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_A(\theta)$

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

$$\pi_{\mathrm{new}}(\mathbf{a}|\mathbf{s}) = \arg\max_\pi E_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] \text{ s.t. } D_{\mathrm{KL}}(\pi \| \pi_\beta) \leq \epsilon$$

Peng*, Kumar*, Levine. **Advantage-Weighted Regression.** '19

Nair, Dalal, Gupta, Levine. **Accelerating Online Reinforcement Learning with Offline Datasets.** '20

Can we also avoid all OOD actions in the Q update?

$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \underbrace{E_{\mathbf{a}' \sim \pi_{\text{new}}}[Q(\mathbf{s}', \mathbf{a}')]}$$
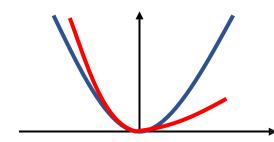
$$V(\mathbf{s}') \longleftarrow \text{ just another neural network}$$

$$V \leftarrow \arg\min_{V} \frac{1}{N} \sum_{i=1}^{N} \ell(V(\mathbf{s}_i), Q(\mathbf{s}_i, \mathbf{a}_i))$$

e.g., MSE loss $(V(\mathbf{s}_i) - Q(\mathbf{s}_i, \mathbf{a}_i))^2$

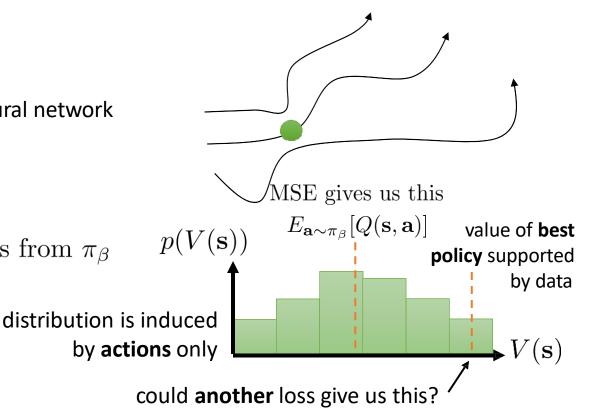this action comes from π_β not from π_{new}

expectile: $\ell_2^\tau(x) = \begin{cases} (1-\tau)x^2 & \text{if } x > 0 \\ \tau x^2 & \text{else} \end{cases}$

$$V(\mathbf{s}) \leftarrow \max_{\mathbf{a} \in \Omega(\mathbf{s})} Q(\mathbf{s}, \mathbf{a})$$

$$\Omega(\mathbf{s}) = \{\mathbf{a} : \pi_\beta(\mathbf{a}|\mathbf{s}) \geq \epsilon\}$$

if we use ℓ_2^τ for big τ

MSE gives us this

$p(V(\mathbf{s}))$ $E_{\mathbf{a} \sim \pi_\beta}[Q(\mathbf{s}, \mathbf{a})]$ value of **best policy** supported by data

distribution is induced by **actions** only

$V(\mathbf{s})$

could **another** loss give us this?

Kostrikov, Nair, Levine. **Offline Reinforcement Learning with Implicit Q-Learning.** '21
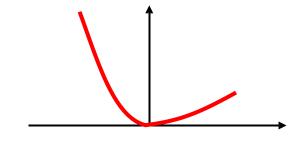
Implicit Q-learning (IQL)

Q-learning with *implicit* policy improvement

$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + V(\mathbf{s}') \qquad V \leftarrow \arg\min_{V} \frac{1}{N} \sum_{i=1}^{N} \ell_2^{\tau}(V(\mathbf{s}_i), Q(\mathbf{s}_i, \mathbf{a}_i))$$

$$V(\mathbf{s}) \leftarrow \max_{\mathbf{a} \in \Omega(\mathbf{s})} Q(\mathbf{s}, \mathbf{a})$$

$$\Omega(\mathbf{s}) = \{\mathbf{a} : \pi_\beta(\mathbf{a}|\mathbf{s}) \geq \epsilon\}$$

if we use ℓ_2^{τ} for big τ

$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \max_{\mathbf{a}' \in \Omega(\mathbf{s}')} Q(\mathbf{s}', \mathbf{a}')$$
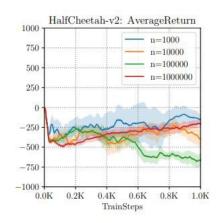
"implicit" policy

$$\pi_{\text{new}}(\mathbf{a}|\mathbf{s}) = \delta(\mathbf{a} = \arg\max_{\mathbf{a} \in \Omega(\mathbf{s})} Q(\mathbf{s}, \mathbf{a}))$$
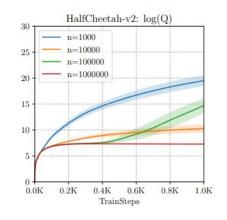
Now we can do value function updates without ever risking out-of-distribution actions!

We'll see results soon, but first let's talk about **Option 2**...

Kostrikov, Nair, Levine. **Offline Reinforcement Learning with Implicit Q-Learning.** '21

Conservative Q-Learning

Conservative Q-learning (CQL)

how well it does

how well it *thinks* it does (Q-values)

$$\hat{Q}^\pi = \arg\min_Q \max_\mu \alpha E_{\mathbf{s}\sim D, \mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})]$$ } term to push down big Q-values

regular objective { $+E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[(Q(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + E_\pi[Q(\mathbf{s}',\mathbf{a}')]))^2\right]$

can show that $\hat{Q}^\pi \leq Q^\pi$ for large enough α

true Q-function

Conservative Q-learning (CQL)

A *better* bound:

always pushes Q-values down   push <u>up</u> on (**s**, **a**) samples in data

$$\hat{Q}^{\pi} = \arg\min_{Q}\max_{\mu}\alpha E_{\mathbf{s}\sim D,\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] -\alpha E_{(\mathbf{s},\mathbf{a})\sim D}[Q(\mathbf{s},\mathbf{a})]$$

$$+E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[(Q(\mathbf{s},\mathbf{a})-(r(\mathbf{s},\mathbf{a})+E_{\pi}[Q(\mathbf{s}',\mathbf{a}')]))^2\right] \Bigg\}\ \mathcal{L}_{\mathrm{CQL}}(\hat{Q}^{\pi})$$

no longer guaranteed that $\hat{Q}^{\pi}(\mathbf{s},\mathbf{a}) \le Q^{\pi}(\mathbf{s},\mathbf{a})$ *for all* $(\mathbf{s},\mathbf{a})$

but guaranteed that $E_{\pi(\mathbf{a}|\mathbf{s})}[\hat{Q}^{\pi}(\mathbf{s},\mathbf{a})] \le E_{\pi(\mathbf{a}|\mathbf{s})}[Q^{\pi}(\mathbf{s},\mathbf{a})]$ *for all* $\mathbf{s} \in D$

Kumar, Zhou, Tucker, Levine. **Conservative Q-Learning for Offline Reinforcement Learning.** '20

Conservative Q-learning (CQL)

1. Update $\hat{Q}^\pi$ w.r.t. $\mathcal{L}_{\mathrm{CQL}}(\hat{Q}^\pi)$ using $\mathcal{D}$

2. Update policy π

if actions are discrete:

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

if actions are continuous:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \sum_i E_{\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i)}[\hat{Q}(\mathbf{s}_i, \mathbf{a})]$$

Kumar, Zhou, Tucker, Levine. **Conservative Q-Learning for Offline Reinforcement Learning.** '20

Conservative Q-learning (CQL)

regularization

$$\hat{Q}^\pi = \arg\min_Q \max_\mu \alpha E_{\mathbf{s}\sim D,\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] - \alpha E_{(\mathbf{s},\mathbf{a})\sim D}[Q(\mathbf{s},\mathbf{a})] - \mathcal{R}(\mu)$$

$$+ E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[\left(Q(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + E_\pi[Q(\mathbf{s}',\mathbf{a}')])\right)^2\right]$$

$$\mathcal{L}_{\mathrm{CQL}}(\hat{Q}^\pi)$$

common choice: $\mathcal{R} = E_{\mathbf{s}\sim D}[\mathcal{H}(\mu(\cdot|\mathbf{s}))]$ maximum entropy regularization

optimal choice: $\mu(\mathbf{a}|\mathbf{s}) \propto \exp(Q(\mathbf{s},\mathbf{a}))$

$$E_{\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] = \log \sum_{\mathbf{a}} \exp(Q(\mathbf{s},\mathbf{a}))$$
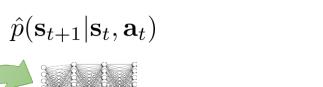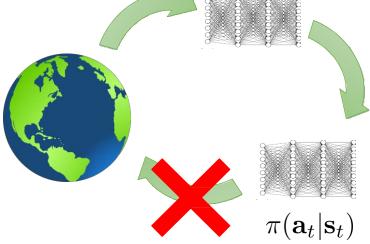
for discrete actions: just calculate directly

for continuous actions: use importance sampling to estimate $E_{\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})]$
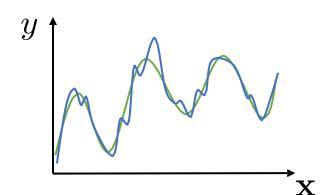
Kumar, Zhou, Tucker, Levine. **Conservative Q-Learning for Offline Reinforcement Learning.** '20

Model-Based Offline RL

How does model-based RL work?

$\hat{p}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$

$\pi(\mathbf{a}_t|\mathbf{s}_t)$

what goes wrong when we can't collect more data?

...so the model's predictions are invalid
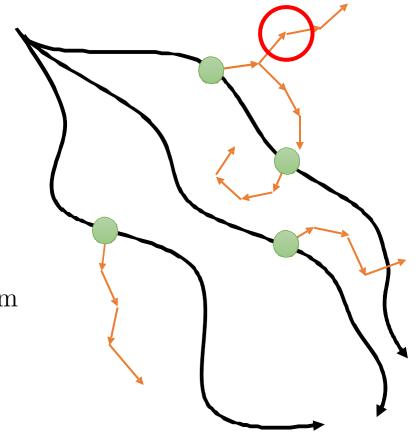
these states are OOD

the model answers "what if" questions

MOPO: Model-Based Offline Policy Optimization

solution: "punish" the policy for exploiting

$$\tilde{r}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) - \lambda u(\mathbf{s}, \mathbf{a})$$

uncertainty penalty

...and then use any existing model-based RL algorithm

Yu*, Thomas*, Yu, Ermon, Zou, Levine, Finn, Ma. **MOPO: Model-Based Offline Policy Optimization.** '20

See also: Kidambi et al., **MOReL : Model-Based Offline Reinforcement Learning.** '20 (concurrent)

MOPO: Theoretical Analysis

$$\tilde{r}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) - \lambda u(\mathbf{s}, \mathbf{a})$$

we can represent the value function

model error is bounded (above) by $u(\mathbf{s}, \mathbf{a})$

Theorem 4.4. *Under Assumption 4.2 and 4.3, the learned policy $\hat{\pi}$ in MOPO (Algorithm 1) satisfies*

true return of policy trained under model ⟶

$$\eta_M(\hat{\pi}) \geq \sup_{\pi} \{\eta_M(\pi) - 2\lambda\epsilon_u(\pi)\} \tag{11}$$

In particular, for all $\delta \geq \delta_{\min}$,

$$\epsilon_u(\pi) := \mathbb{\bar{E}}_{(s,a) \sim \rho_{\hat{T}}^{\pi}}[u(s, a)]$$

$$\eta_M(\hat{\pi}) \geq \eta_M(\pi^{\delta}) - 2\lambda\delta \tag{12}$$

some implications:

$$\pi^{\delta} := \underset{\pi : \epsilon_u(\pi) \leq \delta}{\arg\max} \ \eta_M(\pi)$$

$$\eta_M(\hat{\pi}) \geq \eta_M(\pi^{\mathbf{B}}) - 2\lambda\epsilon_u(\pi^{\mathbf{B}})$$

➢ improves over behavior policy

$$\eta_M(\hat{\pi}) \geq \eta_M(\pi^{\star}) - 2\lambda\epsilon_u(\pi^{\star})$$
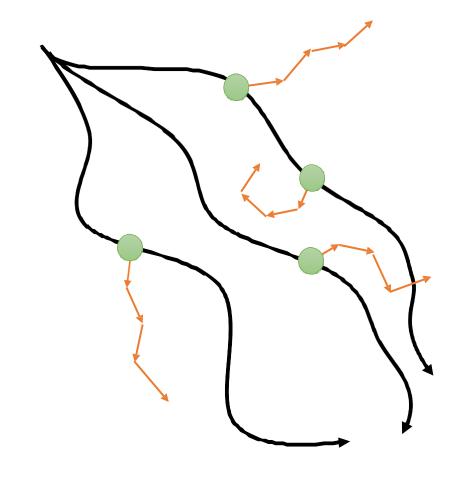
➢ quantifies "optimality gap" in terms of model error

COMBO: Conservative Model-Based RL

Basic idea: just like CQL minimizes Q-value of policy actions, we can minimize Q-value of model state-action tuples

state-action tuples from the model

$$\hat{Q}^{k+1} \leftarrow \arg\min_Q \beta \left(\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\rho(\mathbf{s},\mathbf{a})}[Q(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}[Q(\mathbf{s},\mathbf{a})] \right)$$

$$+ \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim d_f}\left[\left(Q(\mathbf{s},\mathbf{a}) - \widehat{\mathcal{B}}^\pi \hat{Q}^k(\mathbf{s},\mathbf{a}) \right)^2 \right]. \qquad (4)$$

Intuition: if the model produces something that looks clearly different from real data, it's easy for the Q-function to make it look bad

Yu, Kumar, Rafailov, Rajeswaran, Levine, Finn. **COMBO: Conservative Offline Model-Based Policy Optimization**. 2021.
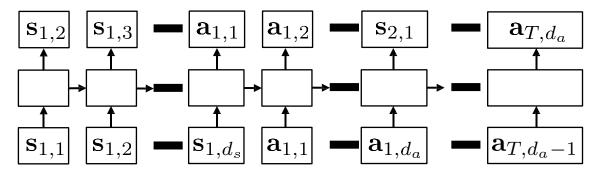
Trajectory Transformer

Basic ideas:

1. train a joint state-action model:

$$p_\beta(\tau) = p_\beta(s_1, a_2, \ldots, s_T, a_T)$$

2. use a big expressive model (a Transformer)

Trajectory Transformer making accurate predictions to hundreds of steps

The model:

How to do control:

beam search, but use $\sum_t r(\mathbf{s}_t, \mathbf{a}_t)$ instead of probability

1. given current sequence, sample next tokens from model
2. store top K tokens with highest cumulative reward
3. move on to next token

Why does this work?

generating high-probability trajectories avoids out-of-distribution states & actions

using a really big model works well in offline mode (lots of compute, captures complex behavior policies)

Janner, Li, Levine. **Reinforcement Learning as One Big Sequence Modeling Problem**. 2021.

Summary, Applications, Open Questions

Which offline RL algorithm do I use?

If you want to *only* train offline…

    Conservative Q-learning   + just one hyperparameter   + well understood and widely tested

    Implicit Q-learning   + more flexible (offline + online)   - more hyperparameters

If you want to *only* train offline and finetune online

    Advantage-weighted actor-critic (AWAC)   + widely used and well tested

    Implicit Q-learning   + seems to perform much better!

If you have a good way to train models in your domain

    COMBO   + similar properties as CQL, but benefits from models

       - not always easy to train a good model in your domain!

    Trajectory transformer   + very powerful and effective models

       - extremely computationally expensive to train and evaluate

The power of offline RL

standard real-world RL process

offline RL process

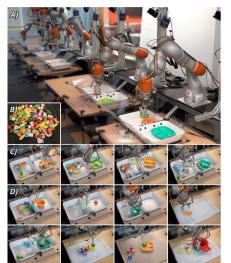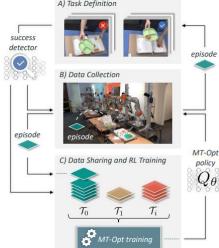# Offline RL in robotic manipulation: MT-Opt, AMs

Kalashnikov, Irpan, Pastor, Ibarz, Herzong, Jang, Quillen, Holly, Kalakrishnan, Vanhoucke, Levine. **QT-Opt: Scalable Deep Reinforcement Learning of Vision-Based Robotic Manipulation Skills**

➢12 different tasks
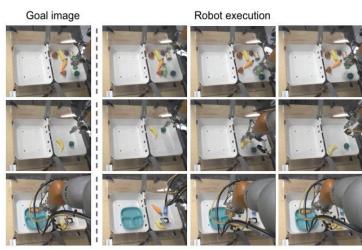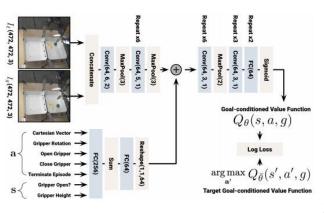➢Thousands of objects
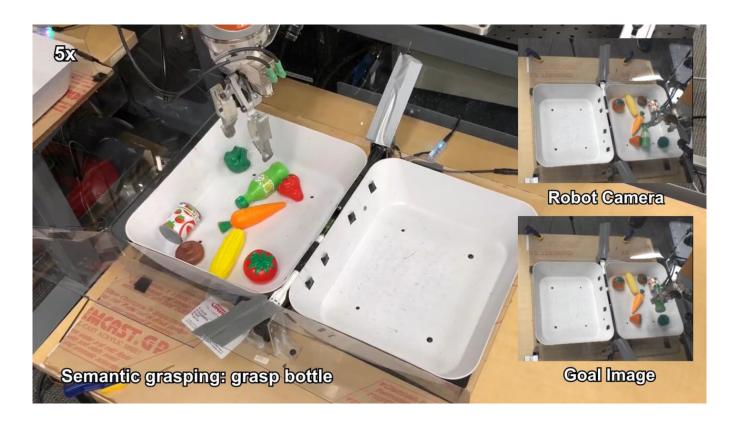➢Months of data collection

reuse the **same exact** data

Kalashnikov, Varley, Chebotar, Swanson, Jonschkowski, Finn, Levine, Hausman. **MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale.** 2021.

New hypothesis: could we learn these tasks **without** rewards using goal-conditioned RL?
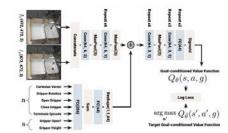
Actionable Models: Offline RL with Goals

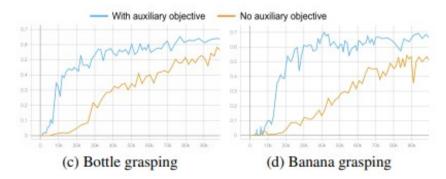

Semantic grasping: grasp bottle

Robot Camera

Goal Image

➢ No reward function at all, task is defined entirely using a **goal image**!
➢ Uses a conservative offline RL method designed for goal-reaching, based on CQL
➢ Works very well as an **unsupervised pretraining** objective!

1. Train **goal-conditioned** Q-function with offline RL

2. Finetune with a **task reward** and limited data

(c) Bottle grasping

(d) Banana grasping

Chebotar, Hausman, Lu, Xiao, Kalashnikov, Varley, Irpan, Eysenbach, Julian, Finn, Levine.
Actionable Models: Unsupervised Offline Reinforcement Learning of Robotic Skills. 2021.

More examples

Early 2020: Greg Kahn collects 40 hours of robot navigation data

Kahn, Abbeel, Levine. **BADGR: An Autonomous Self-Supervised Learning-Based Navigation System.** 2020.

Late 2020: Dhruv Shah uses it to build goal-conditioned navigation system

Demo: Contactless Pizza Delivery

Delivery Location (Image)

in progress
successful

Shah, Eysenbach, Kahn, Rhinehart, Levine. **ViNG: Learning Open-World Navigation with Visual Goals.** 2020.

Early 2021: Dhruv Shah uses the **same** dataset to train an exploration system

Satellite view for visualization purposes only

When deployed in a *previously unseen* environment, RECON explores the environment using a *latent goal model* in search of the target image.

Run 1: Exploration

Target Image

Goal Location

Shah, Eysenbach, Rhinehart, Levine. **RECON: Rapid Exploration for Open-World Navigation with Latent Goal Models.** 2020.

Takeaways, conclusions, future directions

current offline RL algorithms

"the gap"

"the dream"

1. Collect a dataset using any policy or mixture of policies

2. Run offline RL on this dataset to learn a policy

3. Deploy the policy in the real world

- An offline RL **workflow**
  - Supervised learning workflow: train/test split
  - Offline RL workflow: **???**
- Statistical **guarantees**
  - Biggest challenge: distributional shift/counterfactuals
  - Can we make any guarantees?
- Scalable methods, large-scale applications
  - Dialogue systems
  - Data-driven navigation and driving

A starting point: Kumar, Singh, Tian, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning**. CoRL 2021