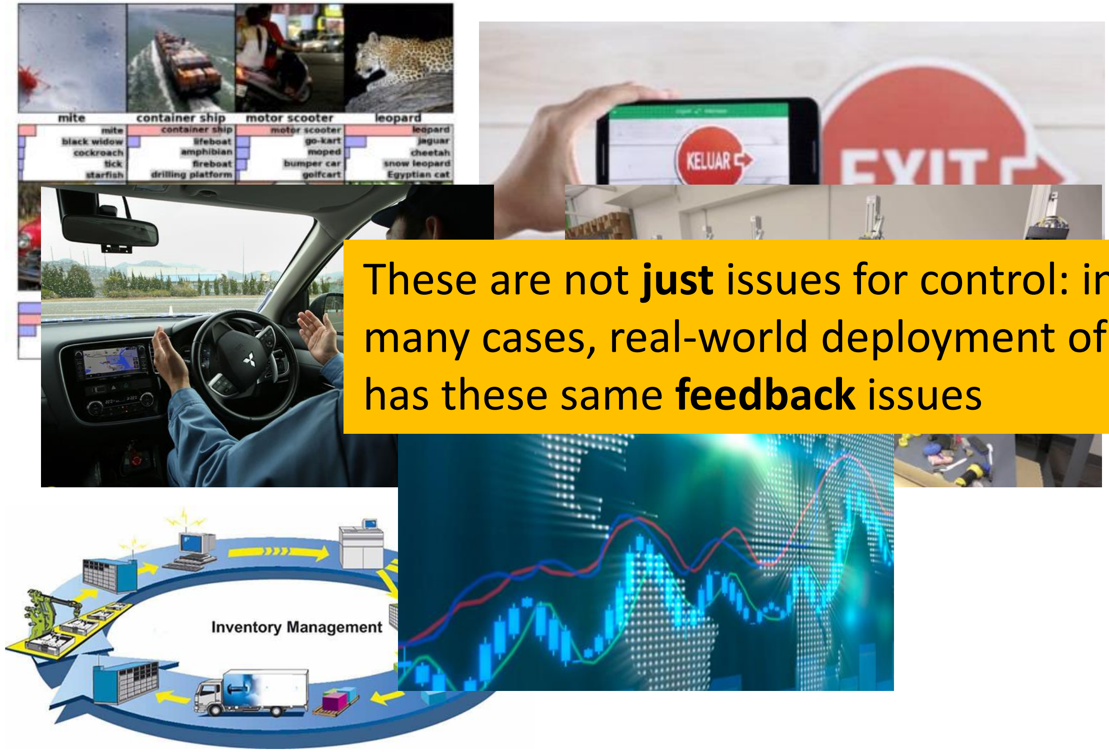
Offline Reinforcement Learning: Representations, Algorithms, and Applications

Sergey Levine

UC Berkeley

Machine learning is automated decision-making



Typical supervised learning problems have assumptions that make them "easy":

- ➢ independent datapoints
- ➢ outputs don't influence future inputs

These are not **just** issues for control: in many cases, real-world deployment of ML has these same **feedback** issues

Example: decisions made by a traffic prediction system might affect the route that people take, which changes traffic

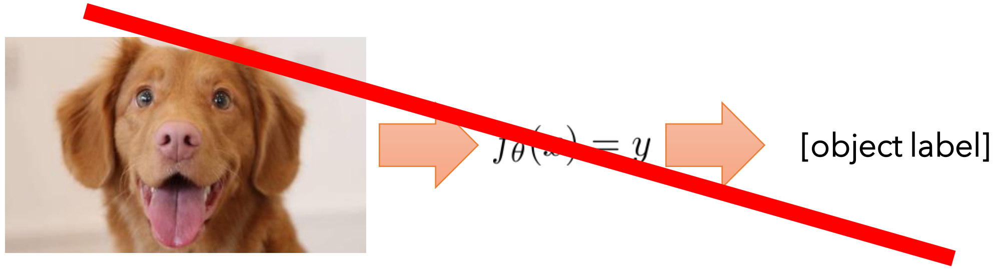
- ➢ current actions influence future observations
- ➢ goal is to maximize some utility (reward)
- ➢ optimal actions are not provided

If ultimately ML is always about making a decision, why don't we treat every machine learning problem like a reinforcement learning problem?

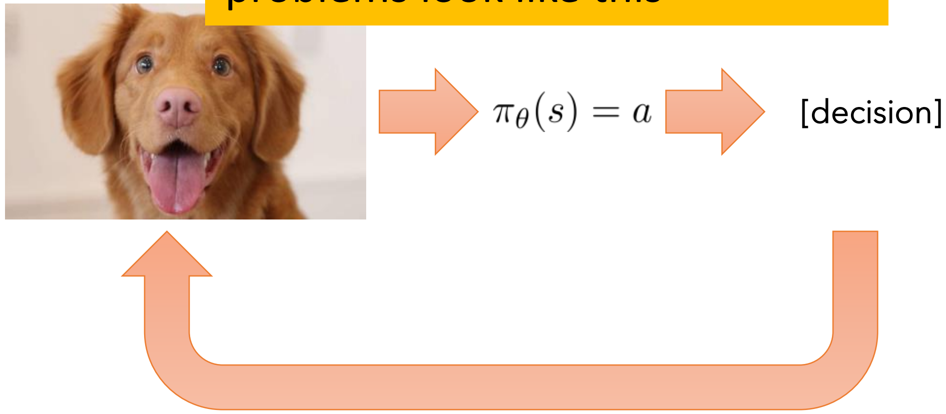
So why aren't we all using RL?

Reinforcement learning is two different things:

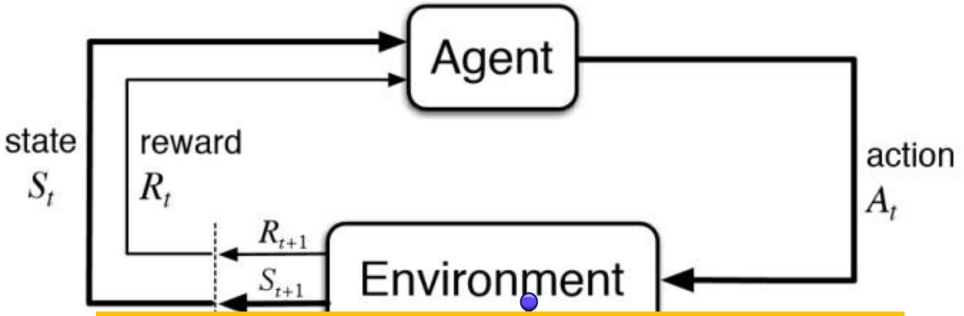
1. Framework for learning-based **decision making**



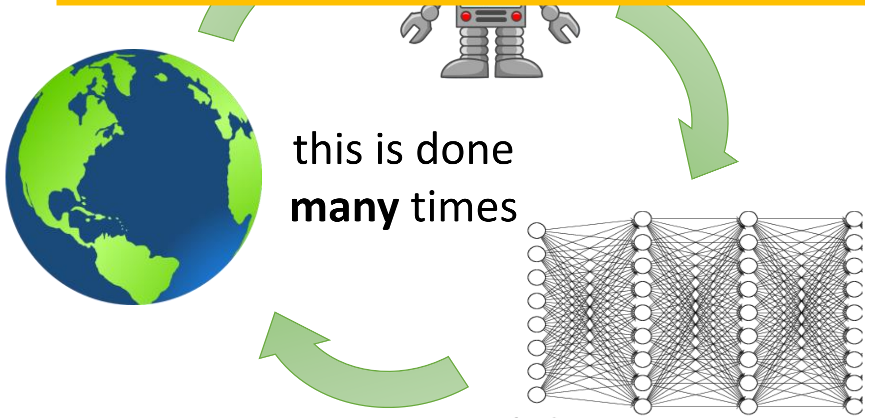
almost all real-world learning problems look like this



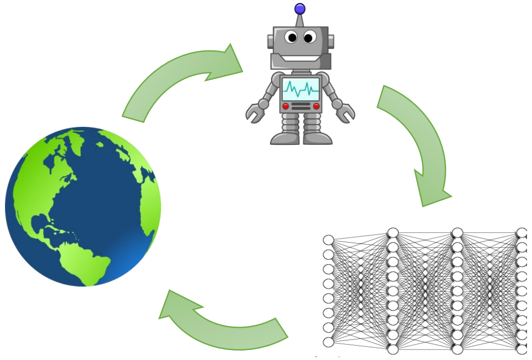
2. **Active**, **online** learning algorithms for control

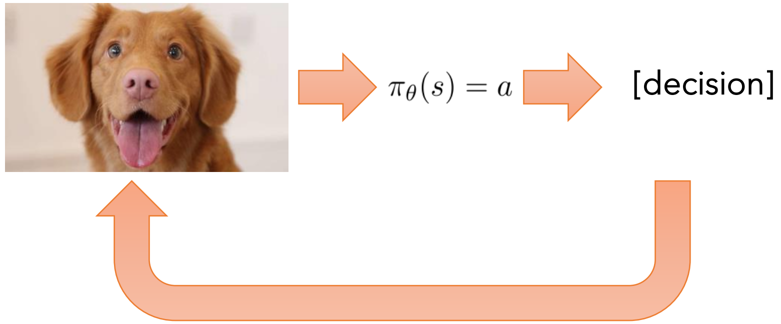


almost all real-world learning problems make it very difficult to do this

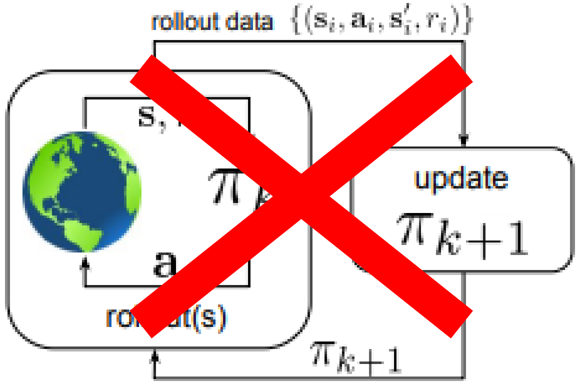


Making RL look more like supervised learning

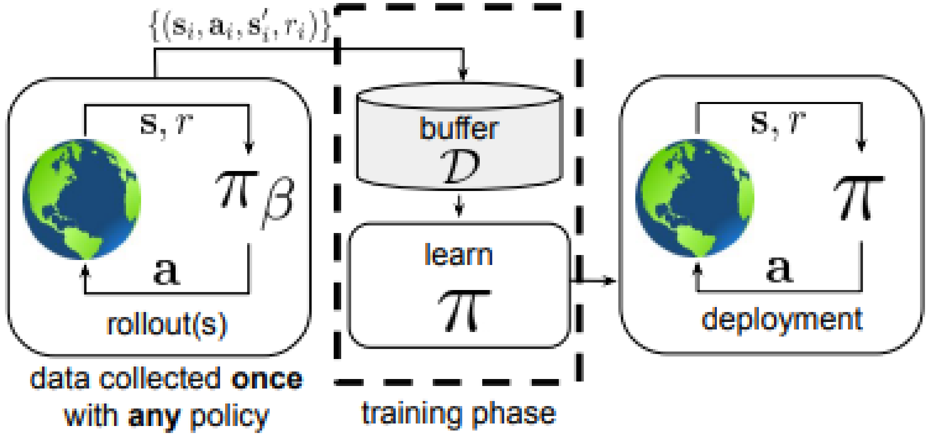


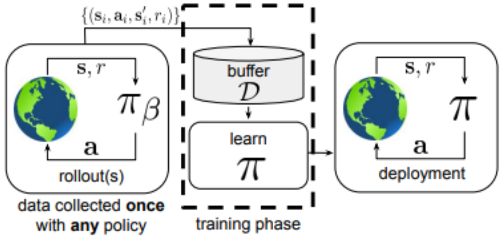


on-policy RL

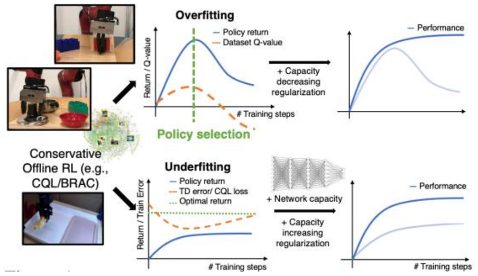


offline reinforcement learning

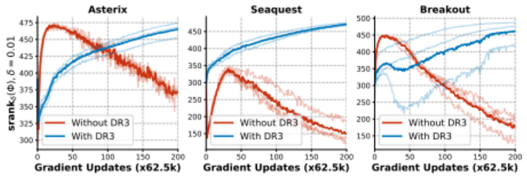




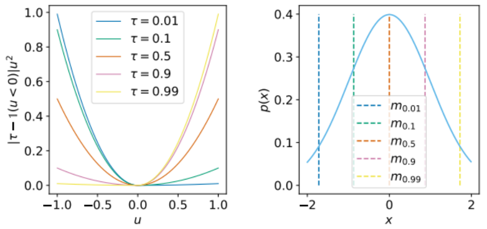
Offline RL challenges & methods



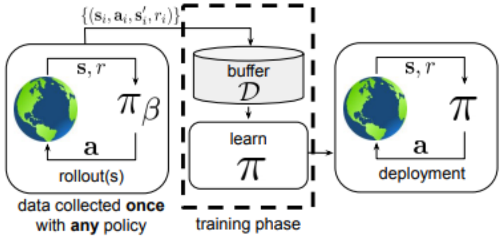
Workflows for offline RL



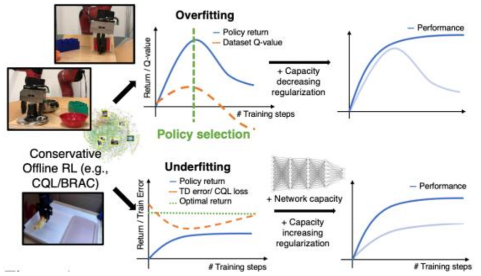
Offline RL and representations



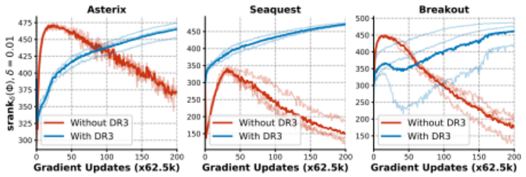
Offline RL without *explicit* pessimism?



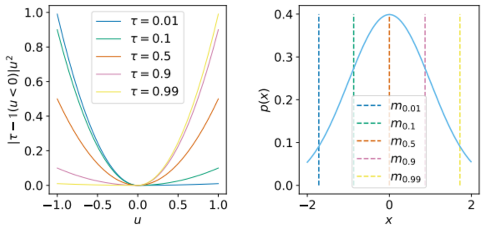
Offline RL challenges & methods



Workflows for offline RL

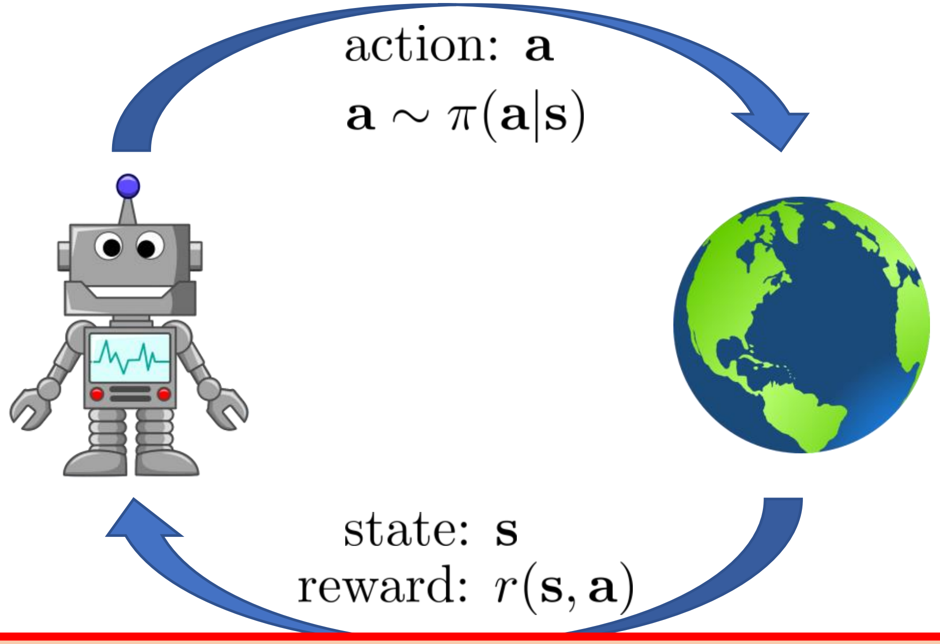


Offline RL and representations



Offline RL without *explicit* pessimism?

Off-policy RL



RL objective: $\max_\pi \sum_{t=1}^{T} E_{\mathbf{s}_t, \mathbf{a}_t \sim \pi}[r(\mathbf{s}_t, \mathbf{a}_t)]$

Q-function: $Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\mathbf{s}_{t'}, \mathbf{a}_{t'} \sim \pi}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t, \mathbf{a}_t]$

$$\pi(\mathbf{a}|\mathbf{s}) = 1 \text{ if } \mathbf{a} = \arg\max_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a})$$

$$Q^\star(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \max_{\mathbf{a}'} Q^\star(\mathbf{s}', \mathbf{a}')$$

enforce this equation at all states!

This talk focuses entirely on **approximate dynamic programming** methods, but there are other methods too!

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

$$\text{minimize } \textstyle\sum_i (Q(\mathbf{s}_i, \mathbf{a}_i) - y_i)^2$$

Why offline RL suffers from distributional shift

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

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

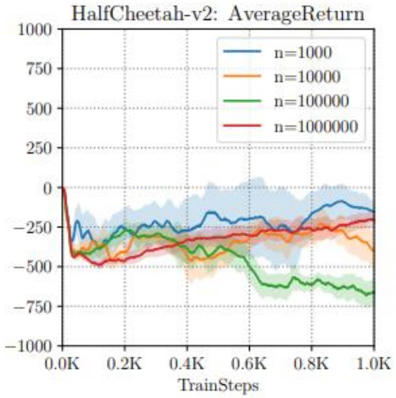
what is the objective?

$$\min_Q \underbrace{E_{(\mathbf{s},\mathbf{a})\sim\pi_\beta(\mathbf{s},\mathbf{a})}}_{\text{behavior policy}} \left[\left(Q(\mathbf{s},\mathbf{a}) - \underbrace{y(\mathbf{s},\mathbf{a})}_{\text{target value}}\right)^2\right]$$

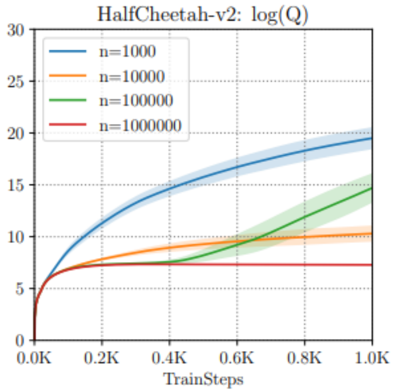
expect good accuracy when $\pi_\beta(\mathbf{a}|\mathbf{s}) = \pi_{\text{new}}(\mathbf{a}|\mathbf{s})$

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

how often does *that* happen?



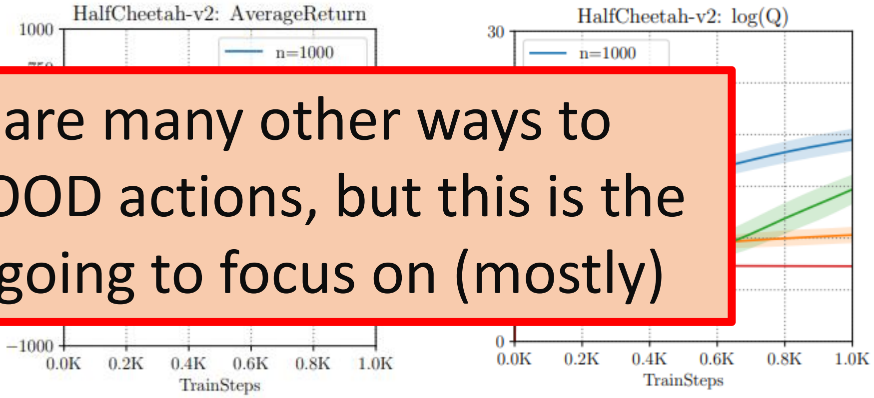
how well it does



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

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

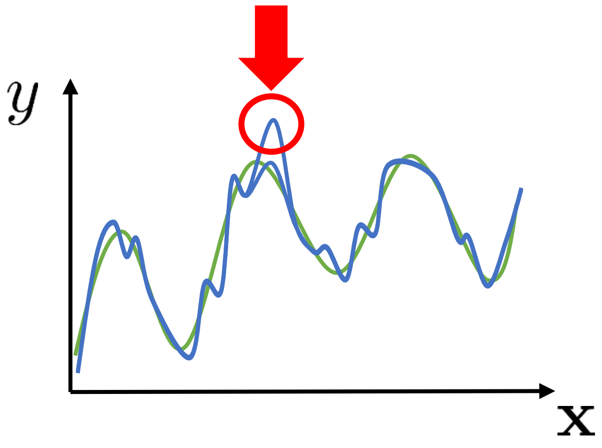
Training the Q-function to avoid OOD errors



There are many other ways to address OOD actions, but this is the one I'm going to focus on (mostly)

how well it does

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



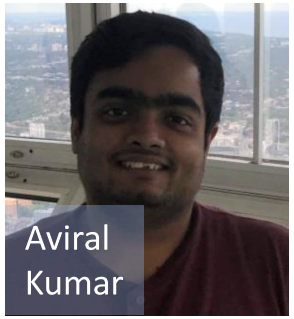
$$\hat{Q}^{\pi} = \arg\min_{Q}\max_{\mu} \alpha E_{\mathbf{s}\sim D,\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})]$$ term to push down big Q-values

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

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

true Q-function

Learning with Q-function lower bounds
Conservative Q-learning (CQL)



A *better* bound:

<u>always</u> pushes Q-values down

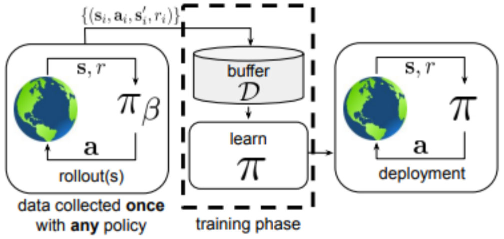
push <u>up</u> on (**s**, **a**) samples in data

$$\hat{Q}^\pi = \arg\min_Q \max_\mu \alpha E_{\mathbf{s}\sim D, \mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] - \alpha E_{(\mathbf{s},\mathbf{a})\sim D}[Q(\mathbf{s},\mathbf{a})]$$
$$+ E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[\left(Q(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + E_\pi[Q(\mathbf{s}',\mathbf{a}')])\right)^2\right]$$

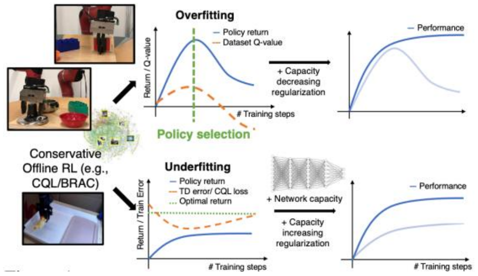
no longer guaranteed that $\hat{Q}^\pi(\mathbf{s},\mathbf{a}) \leq Q^\pi(\mathbf{s},\mathbf{a})$ *for all* $(\mathbf{s},\mathbf{a})$

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

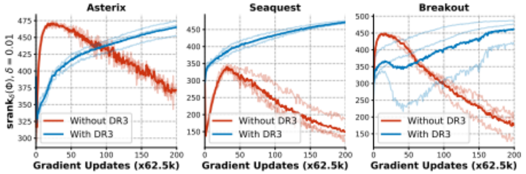
Kumar, Zhou, Tucker, Levine. **Conservative Q-Learning for Offline Reinforcement Learning.** ‘20



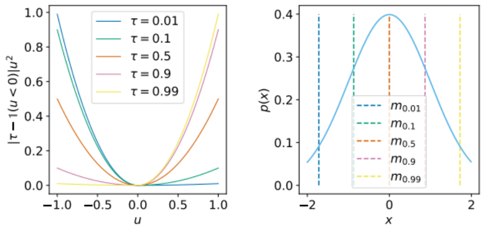
Offline RL challenges & methods



Workflows for offline RL

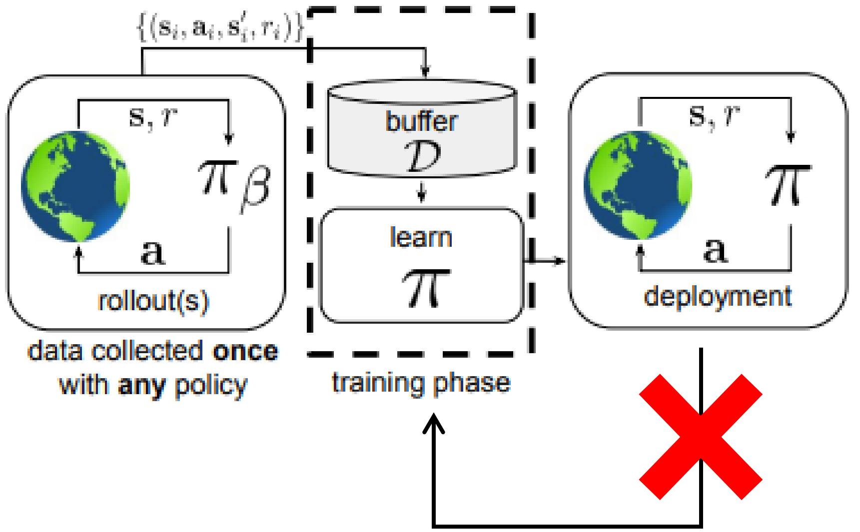


Offline RL and representations



Offline RL without *explicit* pessimism?

The hyperparameter problem



adjust hyperparameters

- **model size**
- **regularization**
- **learning rates**
- **pessimism parameters**

Supervised learning: train/val split

Offline RL: ???

Standard formulation:

off-policy evaluation + model selection

+ very widely studied

- introduces **its own** hyperparameters

- generally a very hard problem

Key observation: to tune hyperparameters, we don't need to evaluate **any** policy, only the policies produced by our specific offline RL method!

Can we leverage properties of a specific offline RL method (e.g., CQL) to develop a **workflow** that allows selecting hyperparameters **without** off-policy evaluation?

“Overfitting” vs. “underfitting”

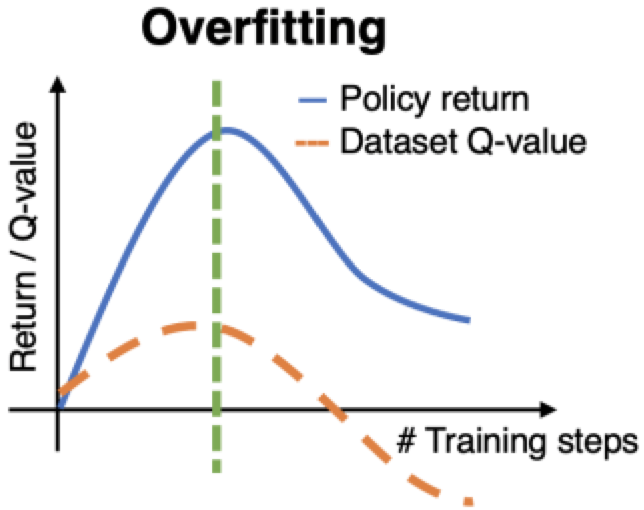
Quantity	Supervised Learning	Conservative Offline RL
Test error	Loss $\mathcal{L}$ evaluated on test data, $\mathcal{D}_{\text{test}}$	Performance of policy, $J(\pi)$
Train error	Loss $\mathcal{L}$ evaluated on train data, $\mathcal{D}_{\text{train}}$	Objective in Equations 2, 1
Overfitting	$\mathcal{L}(\mathcal{D}_{\text{train}})$ low, $\mathcal{L}(\mathcal{D}_{\text{val}})$ high, $\mathcal{D}_{\text{val}}$ is a validation set drawn i.i.d. as $\mathcal{D}_{\text{train}}$	Training objective in Equation 1 is extremely low, low value of $J(\pi)$
Underfitting	high value of train error $\mathcal{L}(\mathcal{D}_{\text{train}})$	Training objective in Equation 1 is extremely high, low value of $J(\pi)$

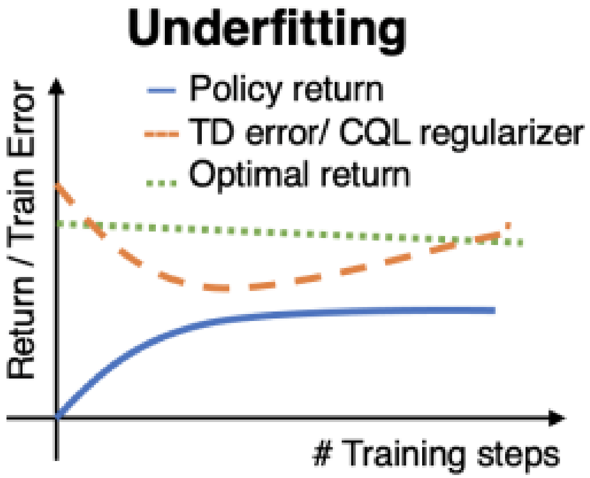
$$\min_{\theta} \alpha \left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\cdot|\mathbf{s})}\left[Q_\theta(\mathbf{s},\mathbf{a})\right] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[Q_\theta(\mathbf{s},\mathbf{a})\right]\right) + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q_\theta(\mathbf{s},\mathbf{a}) - \mathcal{B}^\pi\bar{Q}(\mathbf{s},\mathbf{a})\right)^2\right]$$

(conservative Q-learning)

$$\pi^* := \arg\max_{\pi} \; J_{\mathcal{D}}(\pi) - \alpha D(\pi, \pi_\beta)$$

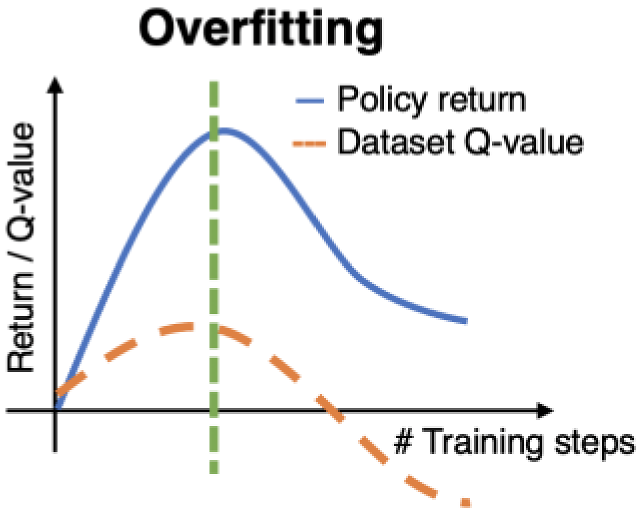
(abstract model of a conservative offline RL method)





Kumar*, Singh*, Tian, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning.** ‘21

Handling **“Overfitting”**



Why?

if overfitting, these become very low

note that $\mu \approx \pi$

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

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

therefore this becomes very low! so this becomes very low!

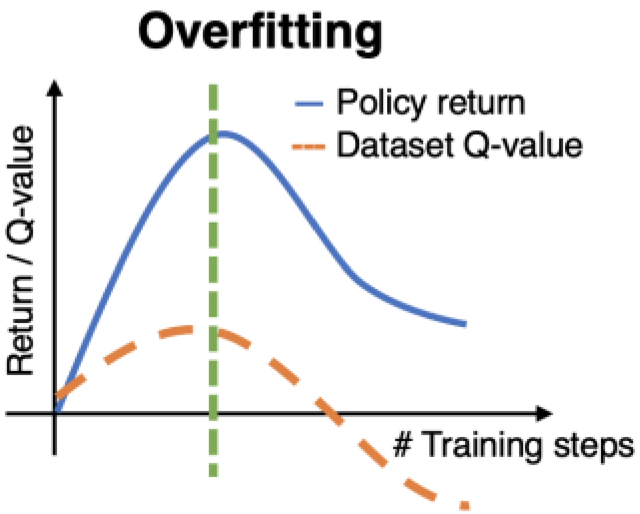
so we get low $E_{(\mathbf{s},\mathbf{a})\sim D}[Q(\mathbf{s},\mathbf{a})]$

If dataset Q-values drop, that means we have too much capacity!

We can fix this by **reducing** capacity or **increasing** regularization

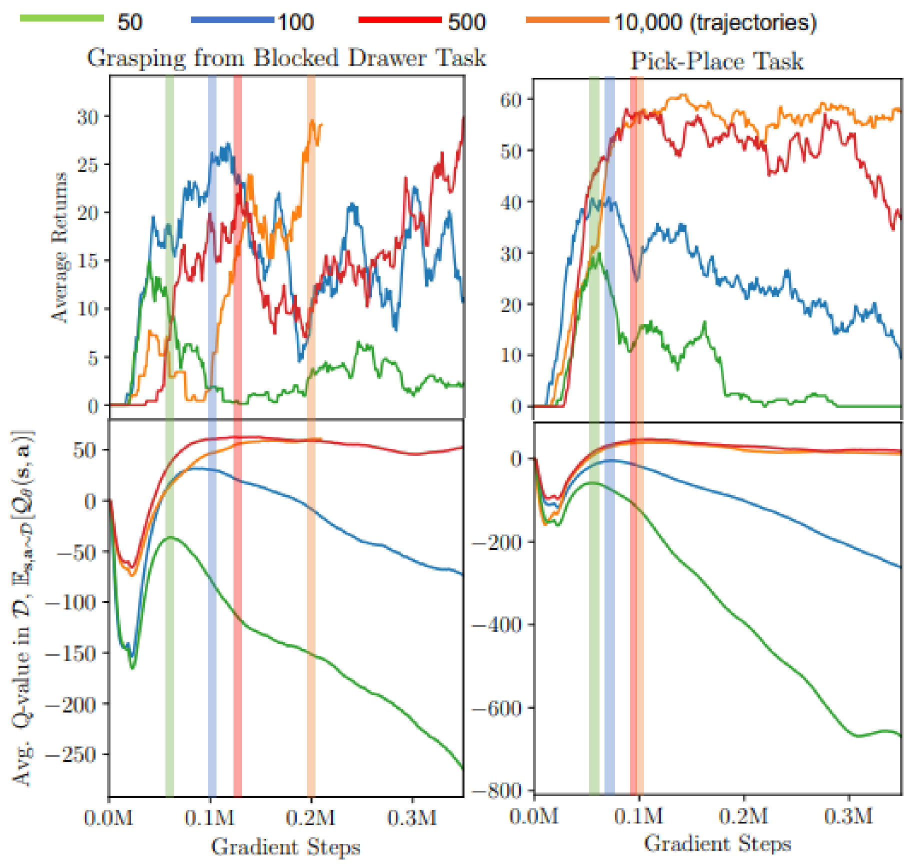
Kumar*, Singh*, Tian, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning.** ‘21

Handling "**Overfitting**"

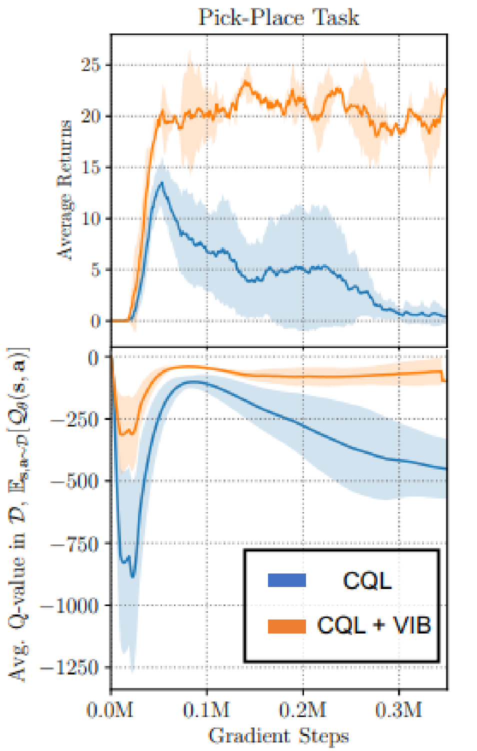


If dataset Q-values drop, that means we have too much capacity!

We can fix this by **reducing** capacity or **increasing** regularization

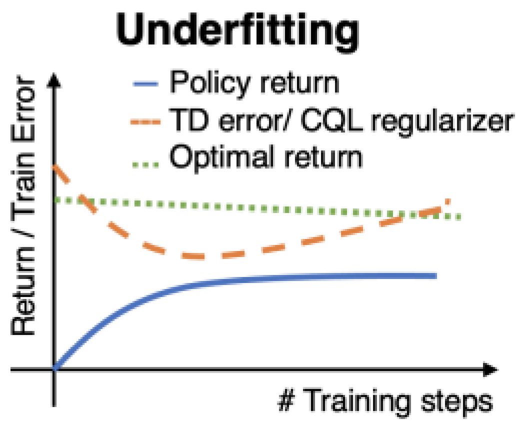


Right: Mitigating Overfitting with VIB



Kumar*, Singh*, Tian, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning.** '21

Handling “Underfitting”



Why?

(so we get **overestimation**)

$$\hat{Q}^\pi = \arg\min_Q \max_\mu \overbrace{\alpha E_{\mathbf{s}\sim D,\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] - \alpha E_{(\mathbf{s},\mathbf{a})\sim D}[Q(\mathbf{s},\mathbf{a})]}^{\text{if underfitting, this is too big}}$$

$$+ \underbrace{E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[\left(Q(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + E_\pi[Q(\mathbf{s}',\mathbf{a}')])\right)^2\right]}_{\text{or this is too big}}$$

Metric 4.2 (Underfitting). *Compute the values of the training TD error, $\mathcal{L}_{\text{TD}}(\theta)$ and CQL regularizer, $\mathcal{R}(\theta)$ for the current run and another identical run with increased model capacity. If the training errors reduce with increasing model capacity, the original run was underfitting.*

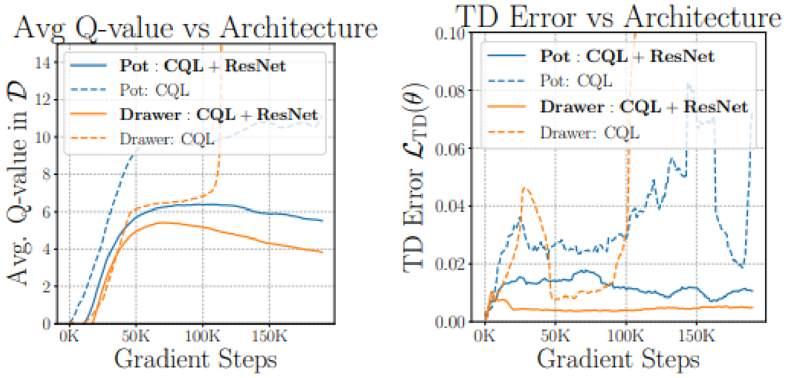
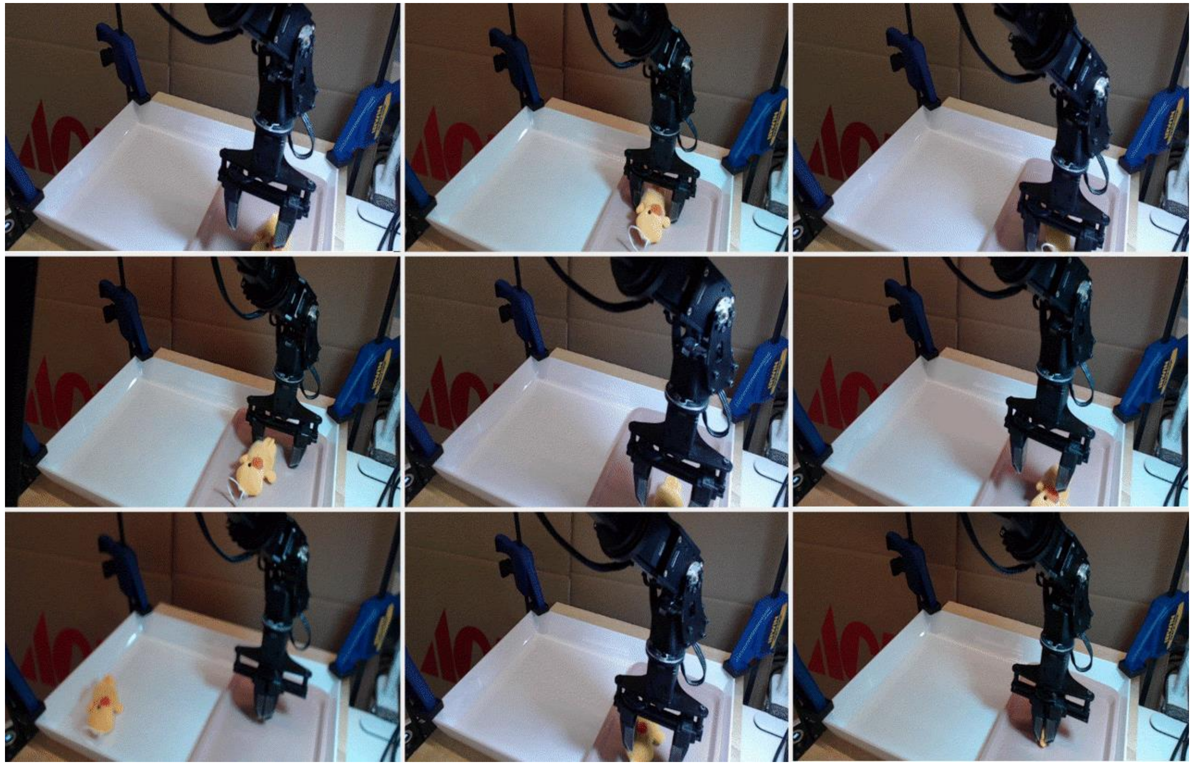


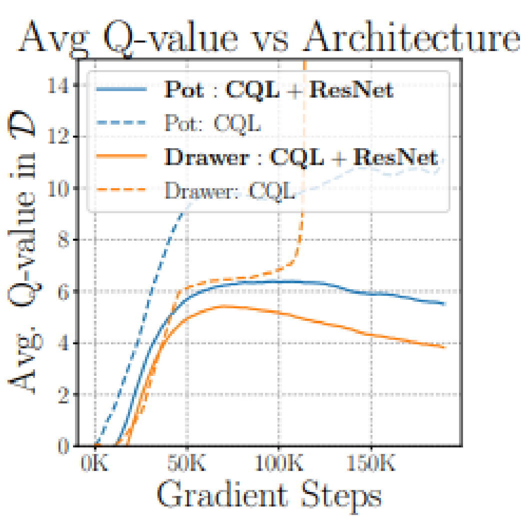
Figure 8: Average Q-value and TD error on Sawyer tasks as model capacity increases. Q-values increase over training with lower capacity ruling out overfitting and increasing model capacity leads to a reduction in TD error indicating the presence of underfitting.

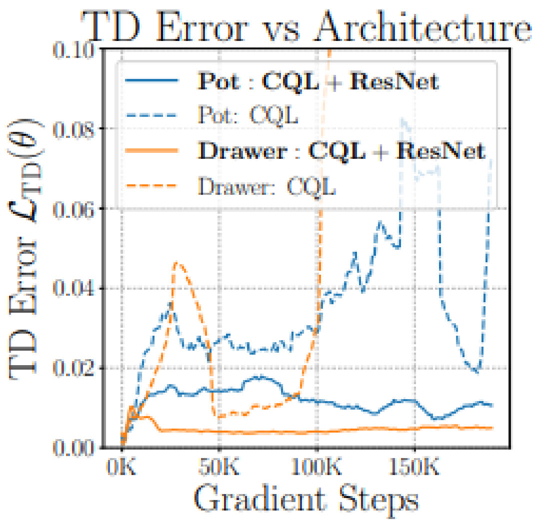
Kumar*, Singh*, Tian, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning.** ‘21

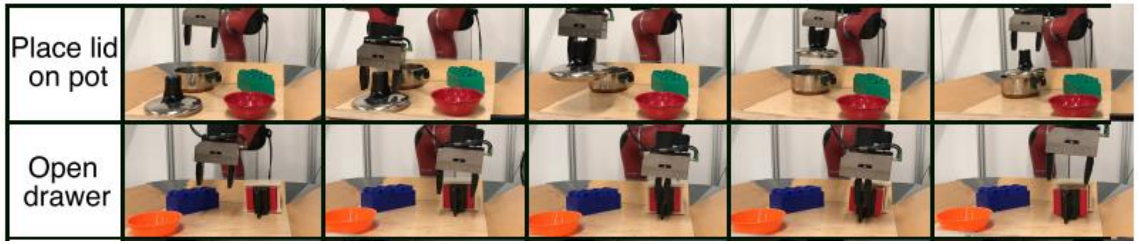
Does it work?

Real-World WidowX Pick and Place: Correcting Overfitting				
Method	Epoch 50	Epoch 75	Epoch 100	Epoch 200
CQL	**7/9**	4/9	4/9	2/9
CQL + VIB	3/9	**8/9**	**7/9**	**7/9**



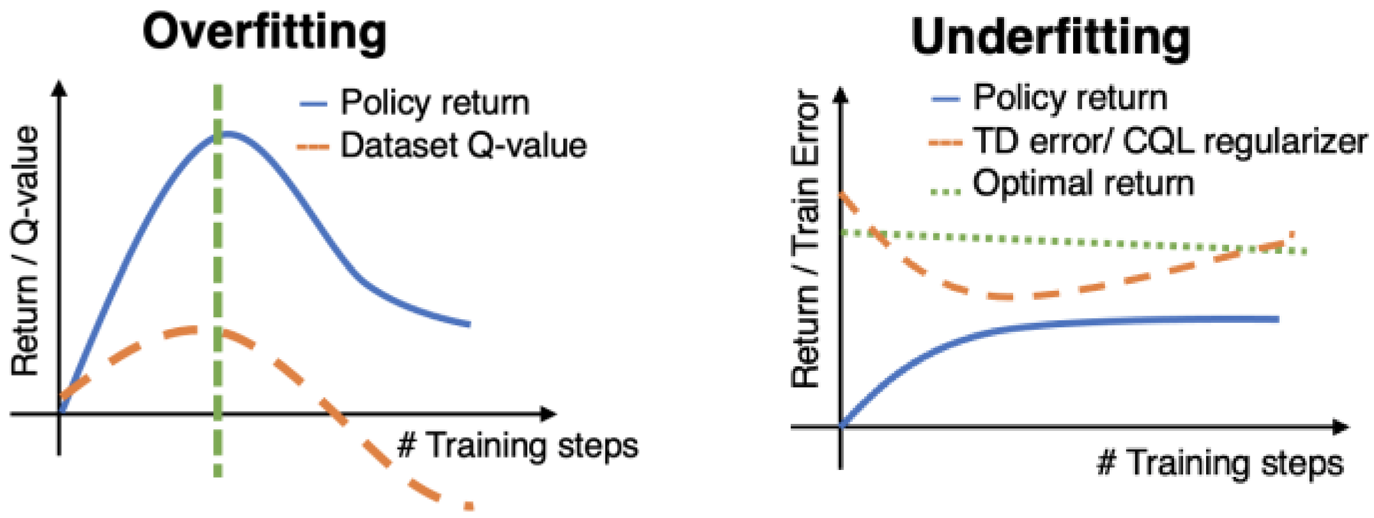






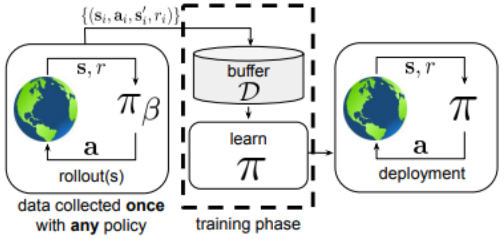
Kumar*, Singh*, Tian, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning.** ‘21

Questions, open problems, opportunities

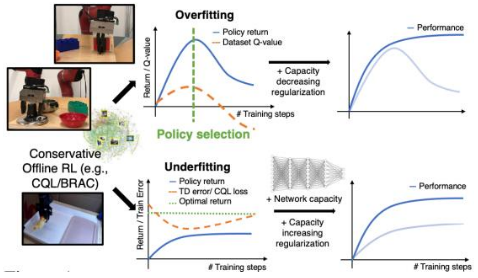


- We have a "workflow" that allows tuning (some) hyperparameters, but doesn't require OPE
- It appears to work in practice, because we can get our robots to work
- It's easier than OPE, because it leverages properties of the corresponding algorithm
- It's rather heuristic
- It's not guaranteed to work every time
- **Can we devise more formally justified, general, and effective workflows?**

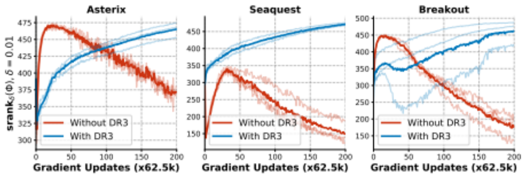
Kumar*, Singh*, Tian, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning.** '21



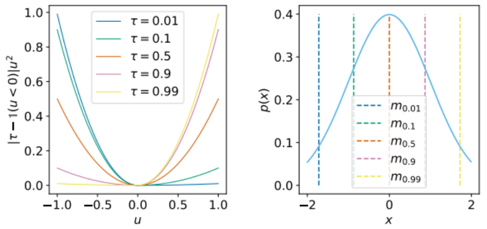
Offline RL challenges & methods



Workflows for offline RL

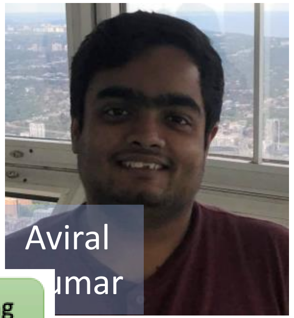


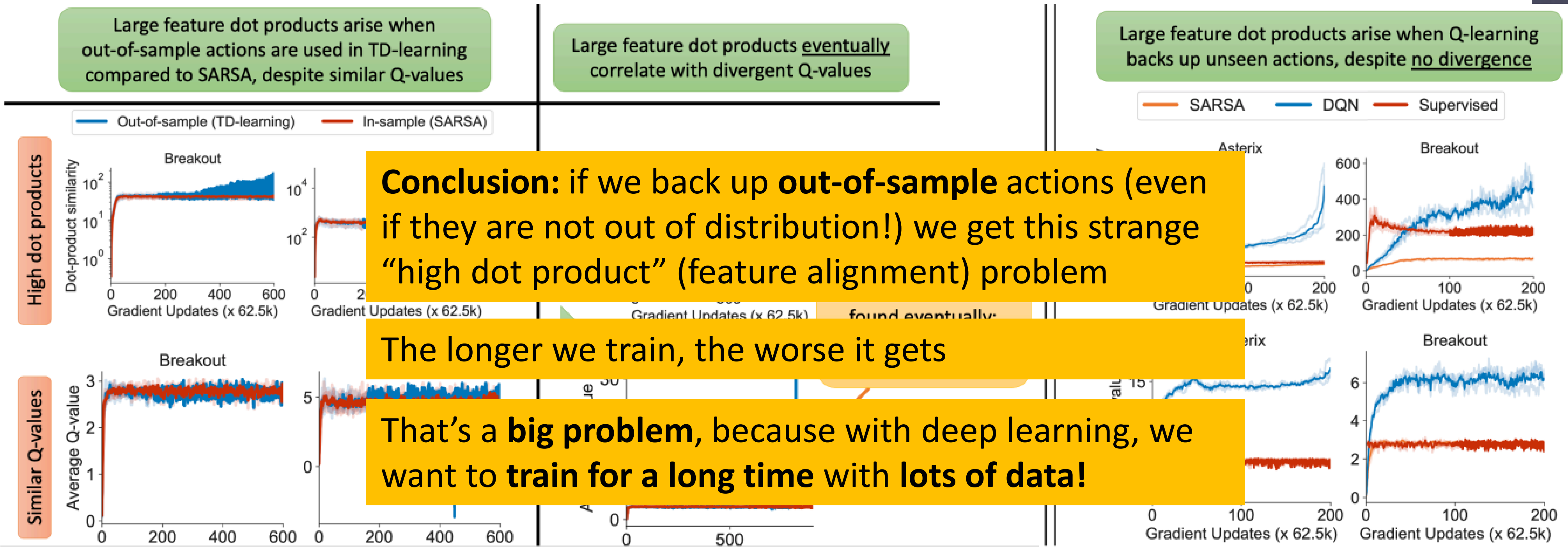
Offline RL and representations



Offline RL without *explicit* pessimism?

The fact that it's a neural network **matters**





Conclusion: if we back up **out-of-sample** actions (even if they are not out of distribution!) we get this strange "high dot product" (feature alignment) problem

The longer we train, the worse it gets

That's a **big problem**, because with deep learning, we want to **train for a long time** with **lots of data!**

high dot product = "aligned" features at consecutive steps

$$\mathbf{s}, \mathbf{a} \rightarrow \quad \phi(\mathbf{s},\mathbf{a})\cdot\phi(\mathbf{s}',\mathbf{a}') \quad \leftarrow \mathbf{s}', \mathbf{a}'$$

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

$Q(\mathbf{s}',\mathbf{a}')$ where $(\mathbf{s}',\mathbf{a}') \in D$ "SARSA"

$Q(\mathbf{s}',\mathbf{a}')$ where $\mathbf{s}' \in D, \mathbf{a}' \sim \pi_\beta(\mathbf{a}'|\mathbf{s}')$ "TD"

Kumar, Agrawal, Tucker, Ma, Levine. **DR3: Value-Based Deep Reinforcement Learning Requires Explicit Regularization.** '21

What's going on?

Blanc et al. (2020); Damian et al. (2021)

Implicit regularization:

$$\theta_{k+1} \leftarrow \theta_k - \eta \nabla_\theta L(\theta) + \eta \varepsilon_k, \;\; \varepsilon_k \sim \mathcal{N}(0, M)$$

if labels corrupted with $\mathcal{N}(0,1)$ noise

$$M = \sum_{i=1}^{|\mathcal{D}|} \nabla_\theta f_\theta(\mathbf{x}_i) \nabla_\theta f_\theta(\mathbf{x}_i)^\top$$

$$R(\theta) = \eta \sum_i^{|\mathcal{D}|} ||\nabla_\theta f_\theta(\mathbf{x}_i)||_2^2$$

when **overparameterized**, solution is stable only if

$$\nabla_\theta R(\theta^*) = 0$$

this is a good thing!

Implicit regularization in reinforcement learning:

Main result: if we follow the TD **pseudo-gradient**

$$\theta_{k+1} = \theta_k - \eta \left(\sum_i \nabla_\theta Q(\mathbf{s}_i, \mathbf{a}_i) \left(Q_\theta(\mathbf{s}_i, \mathbf{a}_i) - (r_i + \gamma Q_\theta(\mathbf{s}'_i, \mathbf{a}'_i)) \right) \right) + \eta \varepsilon_k, \;\; \varepsilon_k \sim \mathcal{N}(0, M)$$

$$R_{\mathrm{TD}}(\theta) = \eta \sum_{i=1}^{|\mathcal{D}|} \underbrace{\nabla Q_\theta(\mathbf{s}_i, \mathbf{a}_i)^\top \Sigma_M^* \nabla Q_\theta(\mathbf{s}_i, \mathbf{a}_i)}_{\text{make gradient inner products small (good)}} - \eta\gamma \sum_{i=1}^{|\mathcal{D}|} \mathrm{trace} \underbrace{\left(\Sigma_M^* \nabla Q_\theta(\mathbf{s}_i, \mathbf{a}_i) \left[[\nabla Q_\theta(\mathbf{s}'_i, \mathbf{a}'_i)^\top] \right] \right)}_{\text{make gradient inner products big (uh oh!)}}$$

make **gradient** inner products small (good) — make **gradient** inner products big (uh oh!)

balances out if $(\mathbf{s}', \mathbf{a}') \in \mathcal{D}$

runaway maximization if $(\mathbf{s}', \mathbf{a}') \notin \mathcal{D}$

Kumar, Agrawal, Tucker, Ma, Levine. **DR3: Value-Based Deep Reinforcement Learning Requires Explicit Regularization.** ‘21

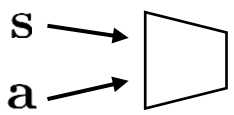
Can we correct this problem?

$$R_{\mathrm{TD}}(\theta) = \eta \sum_{i=1}^{|\mathcal{D}|} \nabla Q_\theta(\mathbf{s}_i, \mathbf{a}_i)^\top \Sigma_M^* \nabla Q_\theta(\mathbf{s}_i, \mathbf{a}_i) - \eta\gamma \sum_{i=1}^{|\mathcal{D}|} \mathrm{trace}\left(\Sigma_M^* \nabla Q_\theta(\mathbf{s}_i, \mathbf{a}_i) \left[\left[\nabla Q_\theta(\mathbf{s}'_i, \mathbf{a}'_i)^\top\right]\right]\right)$$

what if we **add** explicit regularization to balance out the second term?

should be something like $E_{\mathcal{D}}[\nabla Q_\theta(\mathbf{s}_i, \mathbf{a}_i) \cdot \nabla Q_\theta(\mathbf{s}'_i, \mathbf{a}'_i)]$

works, but expensive



$\phi(\mathbf{s}, \mathbf{a}) \cdot \mathbf{w} = Q(\mathbf{s}, \mathbf{a})$

simple hack: at last layer, $\nabla_{\mathbf{w}} Q_\theta(\mathbf{s}, \mathbf{a}) = \phi(\mathbf{s}, \mathbf{a})$

$$E_{\mathcal{D}}[\nabla_{\mathbf{w}} Q_\theta(\mathbf{s}_i, \mathbf{a}_i) \cdot \nabla_{\mathbf{w}} Q_\theta(\mathbf{s}'_i, \mathbf{a}'_i)] = E_{\mathcal{D}}[\underbrace{\phi(\mathbf{s}_i, \mathbf{a}_i) \cdot \phi(\mathbf{s}'_i, \mathbf{a}'_i)}_{\text{cheap \& easy}}]$$

Kumar, Agrawal, Tucker, Ma, Levine. **DR3: Value-Based Deep Reinforcement Learning Requires Explicit Regularization.** '21

Does this help?

Table 1: IQM normalized average performance (training stability) across 17 games, with 95% CIs in parenthesis, after 6.5M gradient steps for the 1% setting and 12.5M gradient steps for the 5%, 10% settings. Individual performances reported in Tables F.4-F.10. DR3 improves the stability over both CQL and REM.

Data	CQL	CQL + DR3	REM	REM + DR3
1%	43.7 (39.6, 48.6)	**56.9** (52.5, 61.2)	4.0 (3.3, 4.8)	**16.5** (14.5, 18.6)
5%	78.1 (74.5, 82.4)	**105.7** (101.9, 110.9)	25.9 (23.4, 28.8)	**60.2** (55.8, 65.1)
10%	59.3 (56.4, 61.9)	**65.8** (63.3, 68.3)	53.3 (51.4, 55.3)	**73.8** (69.3, 78)

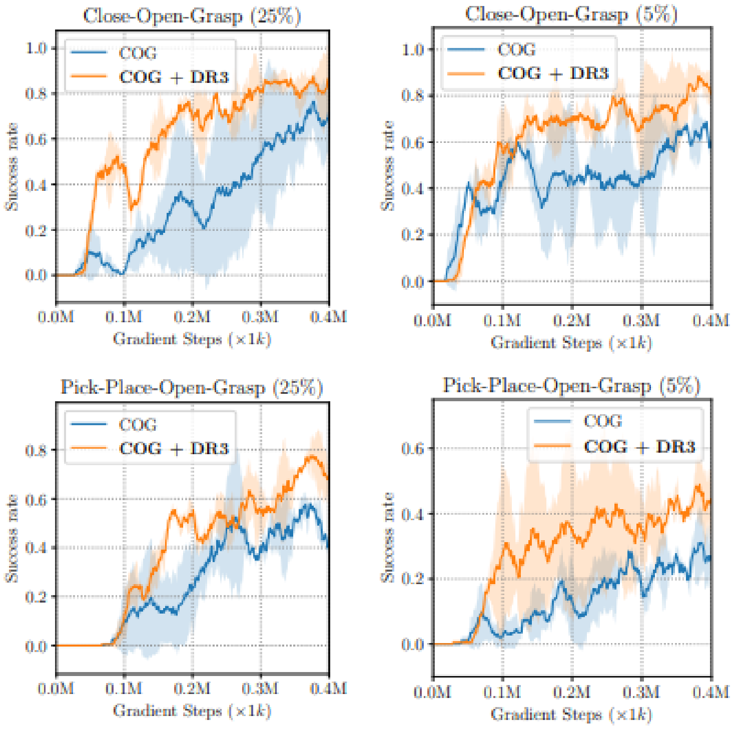


Figure 3: **Performance of DR3 + COG** on two manipulation tasks using only 5% and 25% of the data used by Singh et al. (2020) to make these more challenging. COG + DR3 outperforms COG in training and attains higher average and final performance.

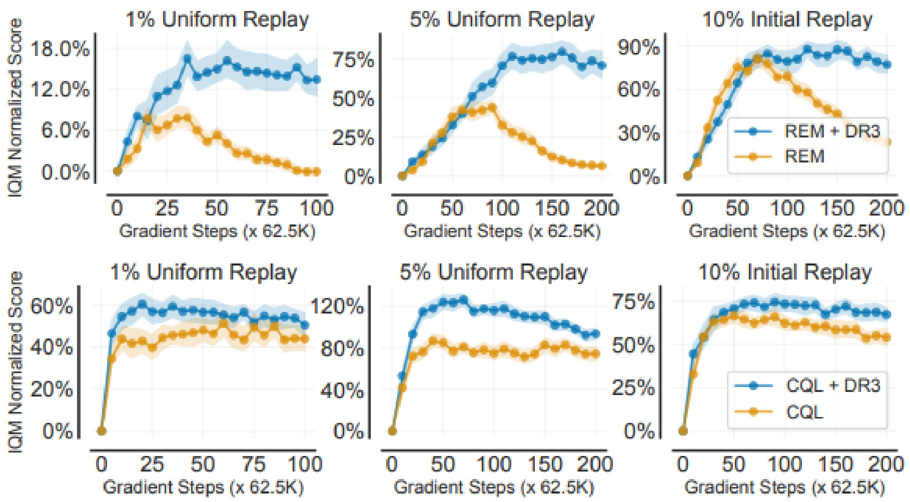
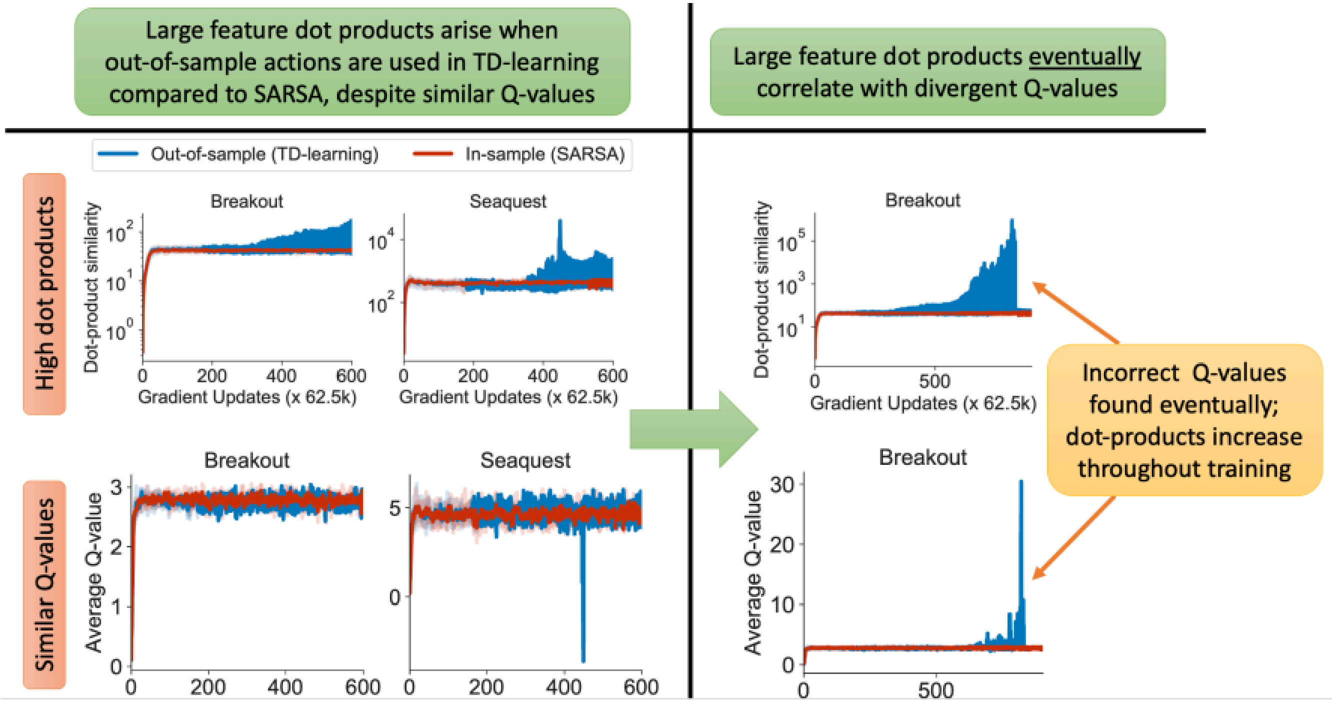


Figure 4: **Normalized performance across 17 Atari games for REM + DR3 (top), CQL + DR3 (bottom).** x-axis represents *gradient steps*; no new data is collected. While naïve REM suffers from a degradation in performance with more training, REM + DR3 not only remains generally stable with more training, but also attains higher final performance. CQL + DR3 attains higher performance than CQL. We report IQM with 95% stratified bootstrap CIs (Agarwal et al., 2021).

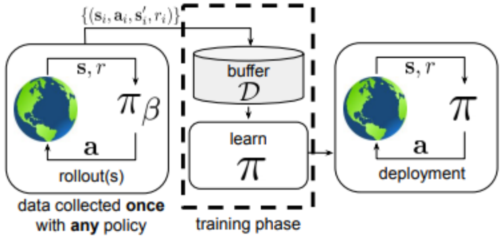
Kumar, Agrawal, Tucker, Ma, Levine. **DR3: Value-Based Deep Reinforcement Learning Requires Explicit Regularization.** ‘21

Conclusions & takeaways

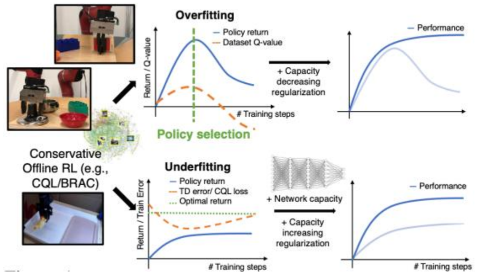
- Offline RL with deep networks (i.e., with representation learning) is fundamentally different from "shallow" RL
- It's also fundamentally different from supervised learning!
- The "usual tricks" that work so well in supervised learning might not lead to great performance in RL directly
- Analyzing the effect of RL training on representations in deep nets is important!



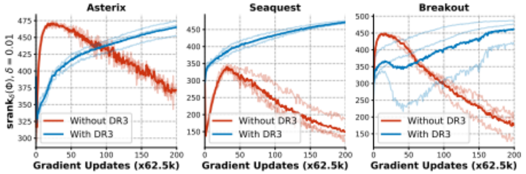
Kumar, Agrawal, Tucker, Ma, Levine. **DR3: Value-Based Deep Reinforcement Learning Requires Explicit Regularization.** '21



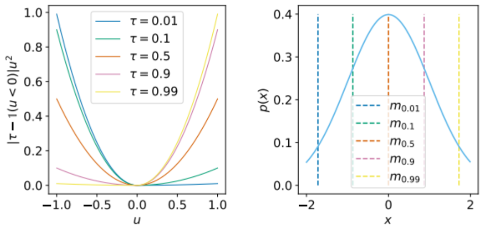
Offline RL challenges & methods



Workflows for offline RL



Offline RL and representations



Offline RL without *explicit* pessimism?

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

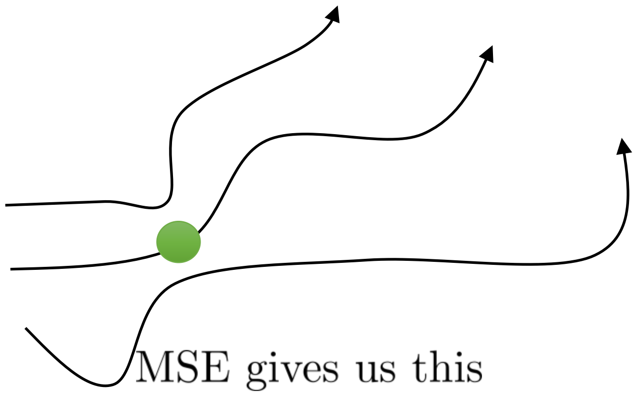
$$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \underbrace{E_{\mathbf{a}'\sim\pi_{\text{new}}}[Q(\mathbf{s}',\mathbf{a}')]}_{V(\mathbf{s}')}$$

$V(\mathbf{s}')$ ← just another neural network

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

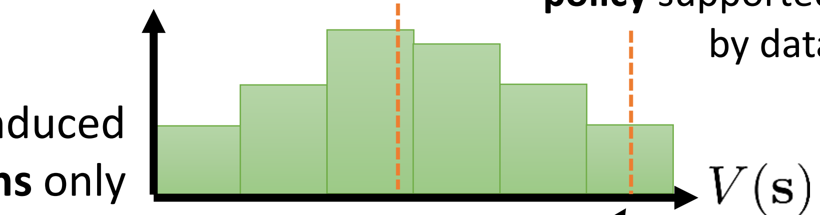
e.g., MSE loss $(V(\mathbf{s}_i) - Q(\mathbf{s}_i,\mathbf{a}_i))^2$

this action comes from π_β not from π_{new}



MSE gives us this $E_{\mathbf{a}\sim\pi_\beta}[Q(\mathbf{s},\mathbf{a})]$

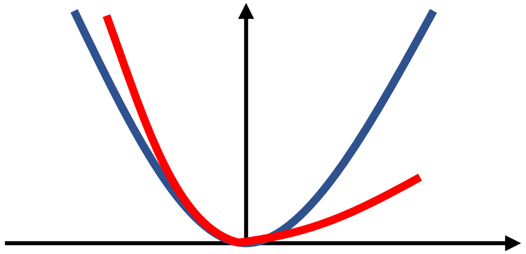
value of **best policy** supported by data



distribution is induced by **actions** only

could **another** loss give us this?

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



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

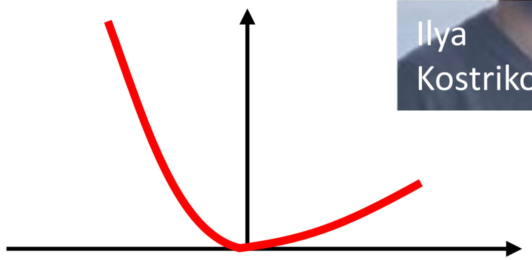
$$\Omega(\mathbf{s}) = \{\mathbf{a} : \pi_\beta(\mathbf{a}|\mathbf{s}) \geq \epsilon\}$$

if we use ℓ_2^τ for big τ

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!

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

Results

Chen et al. Decision Transformers

behavioral cloning best trajectories

behavioral cloning

recent (2021) offline RL methods

Dataset	BC	10%BC	DT	AWAC	Onestep RL	TD3+BC	CQL	IQL
halfcheetah-medium-v2	42.6	42.5	42.6	43.5	48.4	48.3	44.0	47.4
hopper-medium-v2	52.9	56.9	67.6	57.0	59.6	59.3	58.5	66.3
walker2d-medium-v2	75.3	75.0	74.0	72.4	81.8	83.7	72.5	78.3
halfcheetah-medium-replay-v2	36.6	40.6	36.6	40.5	38.1	44.6	45.5	44.2
hopper-medium-replay-v2	18.1	75.9	82.7	37.2	97.5	60.9	95.0	94.7
walker2d-medium-replay-v2	26.0	62.5	66.6	27.0	49.5	81.8	77.2	73.9
halfcheetah-medium-expert-v2	55.2	92.9	86.8	42.8	93.4	90.7	91.6	86.7
hopper-medium-expert-v2	52.5	110.9	107.6	55.8	103.3	98.0	105.4	91.5
walker2d-medium-expert-v2	107.5	109	108.1	74.5	113	110.1	108.8	109.6
locomotion-v2 total	466.7	666.2	672.6	450.7	684.6	677.4	698.5	692.4
antmaze-umaze-v0	54.6	62.8	59.2	56.7	64.3	78.6	74.0	87.5
antmaze-umaze-diverse-v0	45.6	50.2	53.0	49.3	60.7	71.4	84.0	62.2
antmaze-medium-play-v0	0.0	5.4	0.0	0.0	0.3	10.6	61.2	71.2
antmaze-medium-diverse-v0	0.0	9.8	0.0	0.7	0.0	3.0	53.7	70.0
antmaze-large-play-v0	0.0	0.0	0.0	0.0	0.0	0.2	15.8	39.6
antmaze-large-diverse-v0	0.0	6.0	0.0	1.0	0.0	0.0	14.9	47.5
antmaze-v0 total	100.2	134.2	112.2	107.7	125.3	163.8	303.6	378.0
total	566.9	800.4	784.8	558.4	809.9	841.2	1002.1	1070.4
kitchen-v0 total	154.5	-	-	-	-	-	144.6	159.8
adroit-v0 total	104.5	-	-	-	-	-	93.6	118.1
total+kitchen+adroit	825.9	-	-	-	-	-	1240.3	1348.3
runtime	10m	10m	960m	20m	$\approx$ 20m*	20m	80m	20m

most methods get similar results to good BC implementations

significant improvement from methods that properly handle compositionality

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

Finetuning Comparisons

finetunes well, but low offline performance hampers final results

great offline performance, too conservative for finetuning

generally best for finetuning

Dataset	AWAC	CQL	IQL
antmaze-umaze-v0	56.7 → 59.0	70.1 → 99.4	86.7 → 96.0
antmaze-umaze-diverse-v0	49.3 → 49.0	31.1 → 99.4	75.0 → 84.0
antmaze-medium-play-v0	0.0 → 0.0	23.0 → 0.0	72.0 → 95.0
antmaze-medium-diverse-v0	0.7 → 0.3	23.0 → 32.3	68.3 → 92.0
antmaze-large-play-v0	0.0 → 0.0	1.0 → 0.0	25.5 → 46.0
antmaze-large-diverse-v0	1.0 → 0.0	1.0 → 0.0	42.6 → 60.7
antmaze-v0 total	107.7 → 108.3	151.5 → 231.1	370.1 → 473.7
pen-binary-v0	44.6 → 70.3	31.2 → 9.9	37.4 → 60.7
door-binary-v0	1.3 → 30.1	0.2 → 0.0	0.7 → 32.3
relocate-binary-v0	0.8 → 2.7	0.1 → 0.0	0.0 → 31.0
hand-v0 total	46.7 → 103.1	31.5 → 9.9	38.1 → 124.0
total	154.4 → 211.4	182.8 → 241.0	408.2 → 597.7

Option 1: avoid **ever** evaluating actions that are not in the dataset

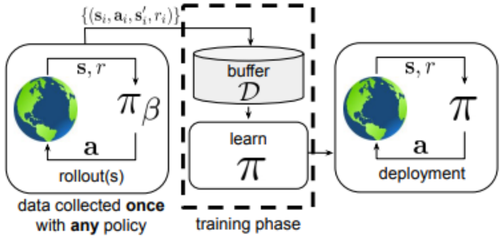
IQL (2021)

Option 2: train the Q-function so that OOD actions never have high values

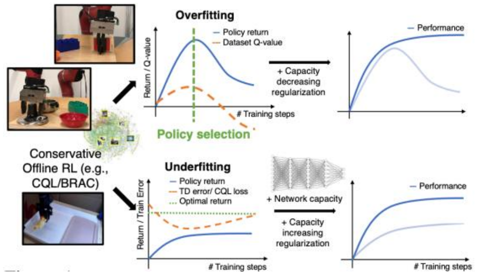
CQL (2020)

- CQL has fewer hyperparameters, cleaner workflows with offline tuning
- CQL has better theoretical guarantees
- IQL performance is slightly better
- IQL finetuning is much better
- We still don't know which principles are going to be more effective in the long run

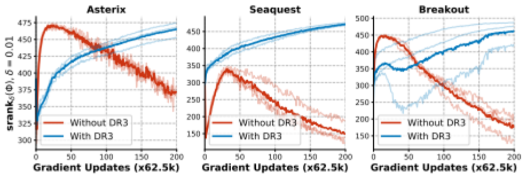
Kostrikov, Nair, Levine. **Offline Reinforcement Learning with Implicit Q-Learning.** '21



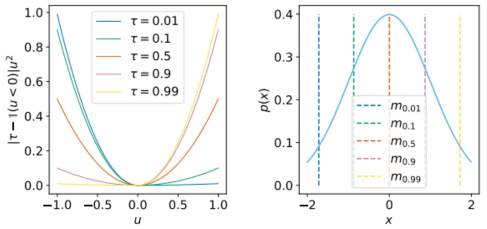
Offline RL challenges & methods



Workflows for offline RL



Offline RL and representations



Offline RL without *explicit* pessimism?

RAIL
Robotic AI & Learning Lab

website: http://rail.eecs.berkeley.edu
source code: http://rail.eecs.berkeley.edu/code.html