Fraunhofer IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS

Reinforcement Learning

—

Lecture 13: Offline Reinforcement Learning

Christopher Mutschler

Offline Reinforcement Learning

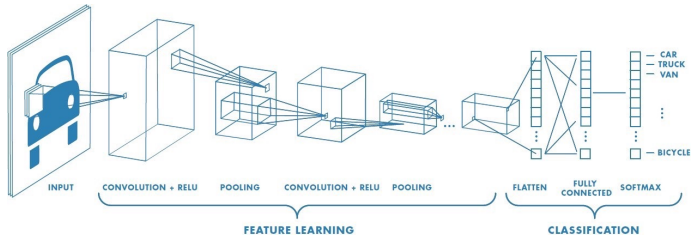
Agenda

- **What is Offline Reinforcement Learning?**
- Challenges of Offline Reinforcement Learning
- Solution Approaches:
  - Policy Constrained Methods:
    - Batch Constrained Q-Learning (BCQ)
    - Bootstrapping Error Accumulation Reduction (BEAR)
  - Conservative Methods:
    - Conservative Q-Learning (CQL)
    - Model-based Offline Policy Optimization (MOPO)
    - Conservative Offline Model-Based Policy Optimization (COMBO)
- Summary

Offline Reinforcement Learning

Motivation

Why is Deep Learning so successful in real-world applications?



Why is this not the same for Reinforcement Learning?

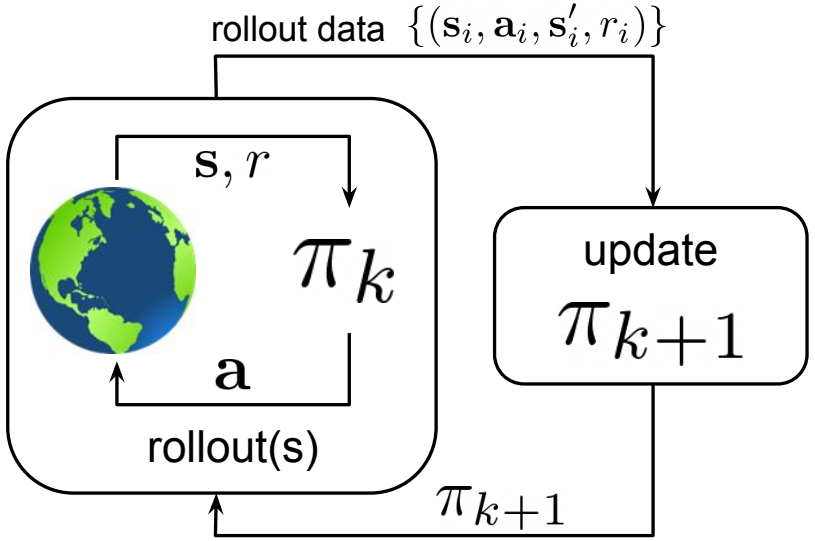
- Active data collection can be costly, impossible or unethical
  - Medical treatment,
  - Autonomous driving, …
- This hinders the application of active reinforcement learning to real world problems
- Simulators are hard to design and the resulting observations inferior to real world data

→ Need for data-driven reinforcement learning that uses passively collected real world data

[1] *http://vladlen.info/projects/scene-understanding/*
[2] *https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53*

Offline Reinforcement Learning

Online (On-Policy) Reinforcement Learning

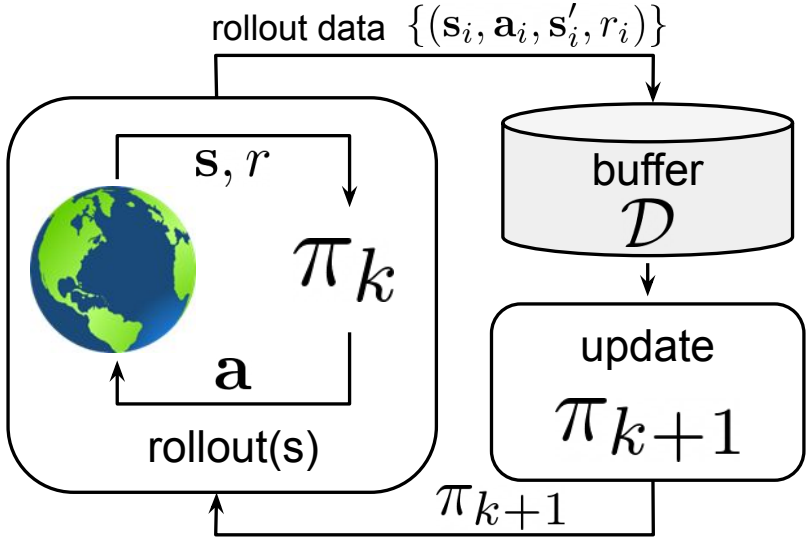


- π_k is updated with streaming data collected by π_k itself
- The transitions (s, a, r, s') are collected using the current policy π_k
- Policy and transitions are perfectly related

Levine et al.: "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems"

Offline Reinforcement Learning

Off-Policy Reinforcement Learning

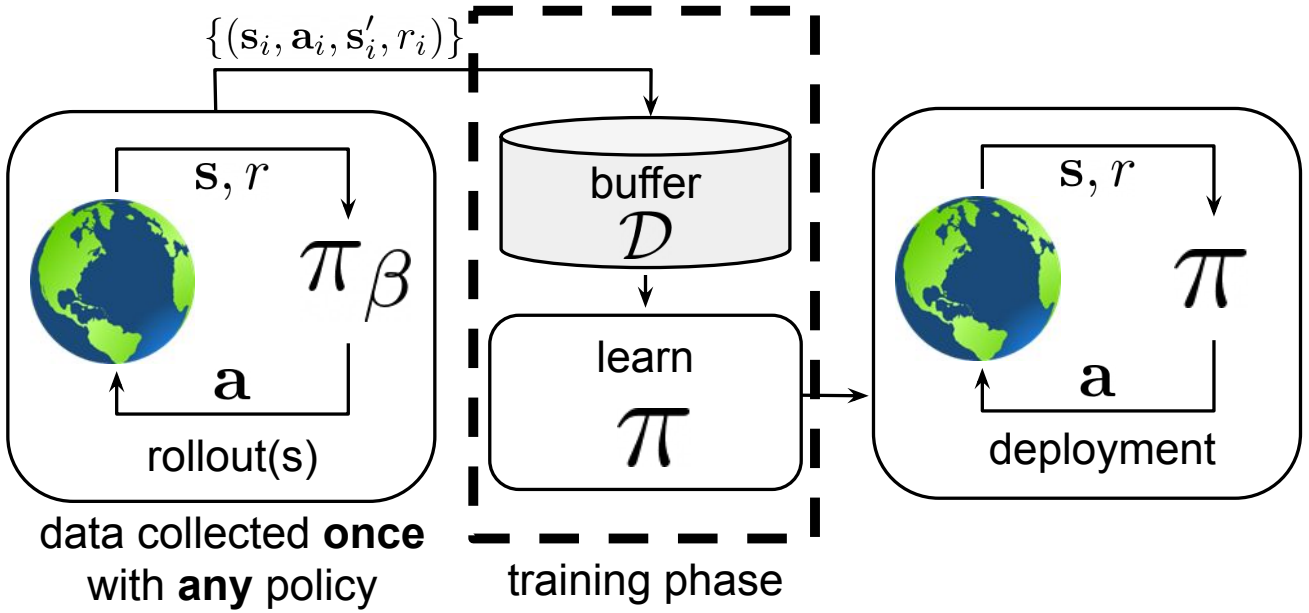


- New transitions (s, a, r, s') are appended to a buffer $\mathcal{D}$
  - $\mathcal{D}$ consist of samples collected under the policies $\pi_0, \pi_1, \ldots, \pi_k$ (with, e.g., an ϵ-greedy sampling)
- All the data from $\mathcal{D}$ is used to train an updated policy π_{k+1} (i.e., transitions are sampled from $\mathcal{D}$)
- Policy and transitions are dependent (strength depends on the size of $\mathcal{D}$ and ϵ)

Levine et al.: "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems"

Offline Reinforcement Learning

Fully Offline Reinforcement Learning



- Offline RL uses a dataset $\mathcal{D}$ collected by some behavior policy π_β
  - π_β is potentially (or often assumed to be) unknown
- $\mathcal{D}$ is **collected once** and **not changed** during training
  - Transitions are sampled from $\mathcal{D}$
  - No interaction with the MDP; Policy is deployed after being fully trained.
- Policy and transitions are independent

Levine et al.: "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems"

Offline Reinforcement Learning

Offline Reinforcement Learning vs. Behavioural Cloning

Behavioural Cloning

- Scenario: Data $\mathcal{D}$ collected from a behavioural agent π_β
- Goal: Learn policy π_β from $\mathcal{D}$ with supervised learning
  1. Needs expert data to get a good policy
  2. Unexpectedly high error bound due to generalization that usually leads to poor performance in practice

Theorem 2.1 (Behavioral cloning error bound). *If $\pi(\mathbf{a}|\mathbf{s})$ is trained via empirical risk minimization on $\mathbf{s} \sim d^{\pi_\beta}(\mathbf{s})$ and optimal labels $\mathbf{a}^\star$, and attains generalization error ϵ on $\mathbf{s} \sim d^{\pi_\beta}(\mathbf{s})$, then $\ell(\pi) \leq C + H^2\epsilon$ is the best possible bound on the expected error of the learned policy.*

Proof. The proof follows from Theorem 2.1 from Ross et al. (2011) using the 0-1 loss, and the bound is the best possible bound following the example from Ross and Bagnell (2010). □

Interestingly, if we allow for additional data collection, where we follow the learned policy $\pi(\mathbf{a}|\mathbf{s})$ to gather additional states $\mathbf{s} \sim d^{\pi}(\mathbf{s})$, and then access optimal action labels for these new *on-policy* states, the best possible bound becomes substantially better:

Theorem 2.2 (DAgger error bound). *If $\pi(\mathbf{a}|\mathbf{s})$ is trained via empirical risk minimization on $\mathbf{s} \sim d^{\pi}(\mathbf{s})$ and optimal labels $\mathbf{a}^\star$, and attains generalization error ϵ on $\mathbf{s} \sim d^{\pi}(\mathbf{s})$, then $\ell(\pi) \leq C + H\epsilon$ is the best possible bound on the expected error of the learned policy.*

Proof. The proof follows from Theorem 3.2 from Ross et al. (2011). This is the best possible bound, because the probability of a mistake at any time step is at least ϵ, and $\sum_{t=1}^{H} \epsilon = H\epsilon$. □

Levine et al.: Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems.

Offline Reinforcement Learning

- Scenario: Data $\mathcal{D}$ collected from a behavioural agent π_β
- Goal: Learn the best possible policy π using only data $\mathcal{D}$

→ Does not necessarily need expert data

Offline Reinforcement Learning

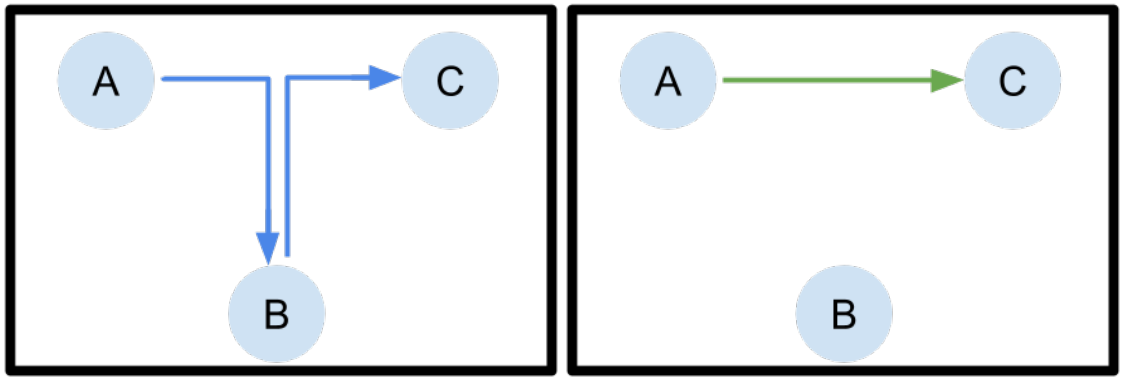
Agenda

- What is Offline Reinforcement Learning?
- **Challenges of Offline Reinforcement Learning**
- Solution Approaches:
  - Policy Constrained Methods:
    - Batch Constrained Q-Learning (BCQ)
    - Bootstrapping Error Accumulation Reduction (BEAR)
  - Conservative Methods:
    - Conservative Q-Learning (CQL)
    - Model-based Offline Policy Optimization (MOPO)
    - Conservative Offline Model-Based Policy Optimization (COMBO)
- Summary

Offline Reinforcement Learning

Why Offline RL Should Work in Principle

- Find the good behaviour in a dataset of good and bad behaviour
- Generalize to similar states and actions
- Stich together good and bad behaviour



https://bair.berkeley.edu/blog/2020/06/25/D4RL/

D4RL:AntMaze

https://sites.google.com/view/d4rl/

Offline Reinforcement Learning

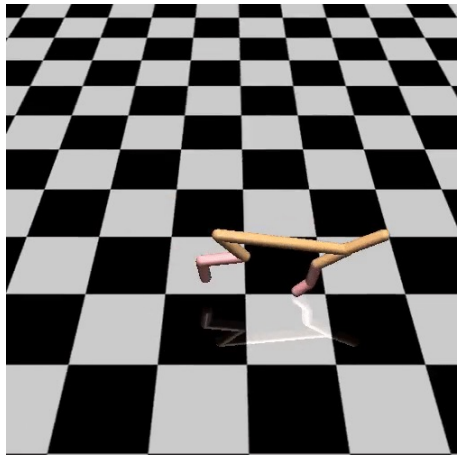
Challenges of Offline Reinforcement Learning

Question

- Can we just use off-policy methods to learn from static datasets?

Experimental Setup

- Train Soft Actor-Critic (SAC) in a fully off-policy setting
- Environment: Mujoco's HalfCheetah-v2
- Use a trained policy to generate *expert demonstrations*
  - Dataset contains successful task completions



https://bair.berkeley.edu/blog/2019/12/05/bear/

Offline Reinforcement Learning

Off-Policy Algorithms for Offline RL?

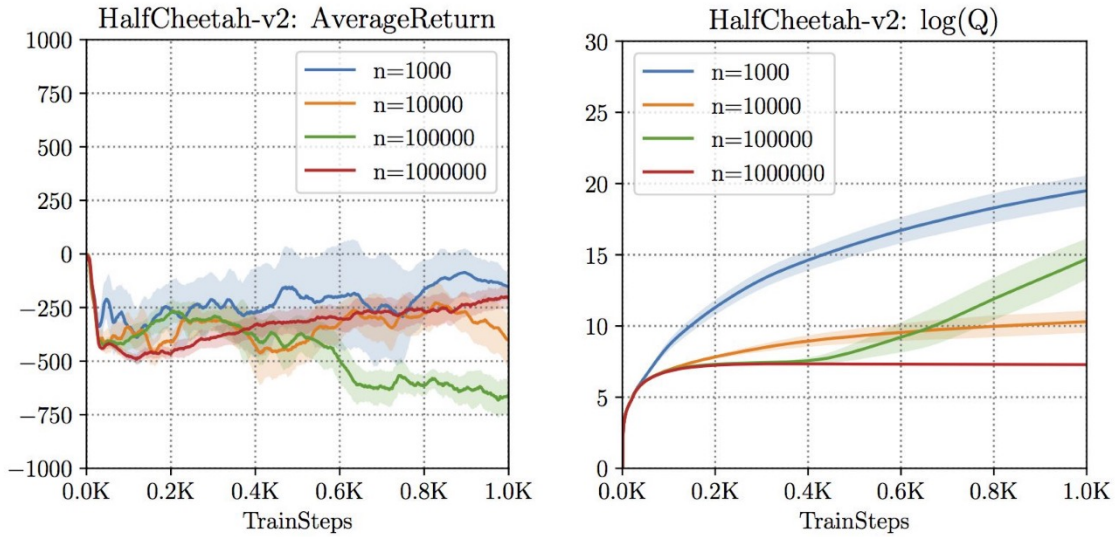
Question

- Can we just use off-policy methods to learn from static datasets? **No!**

Experimental results

- n = size of dataset
- None of the runs succeeds
- Evaluation performance deteriorates with more training (green vs. blue/orange)
  - but cannot be the main reason (see red curve)
- Q-values even diverge for some datasets

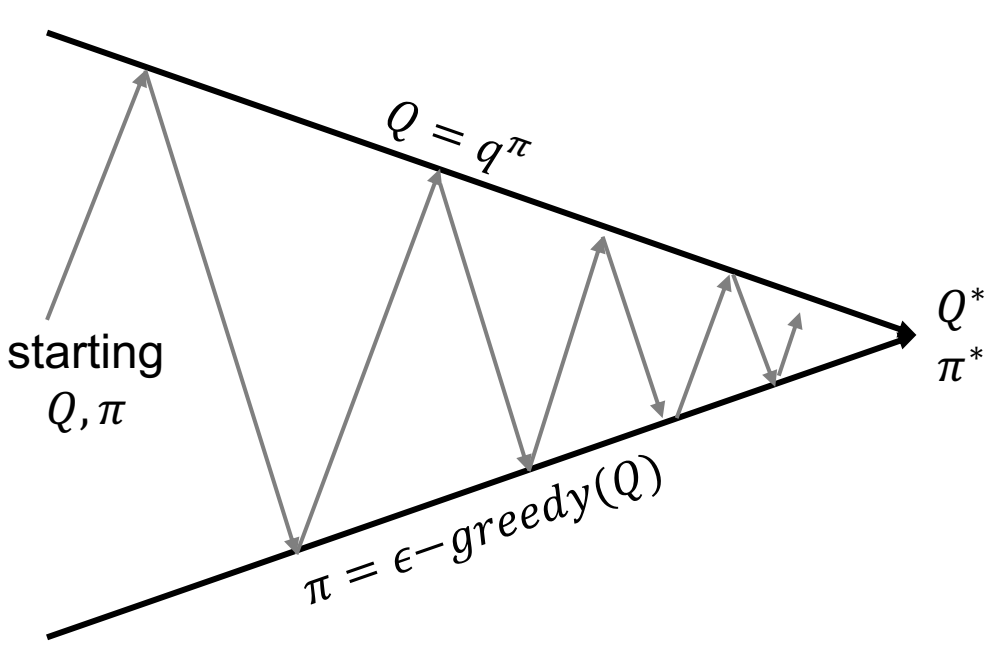
→ **Then why does it fail?**

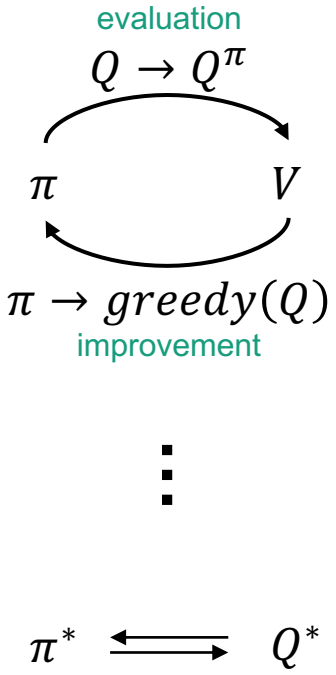


https://bair.berkeley.edu/blog/2019/12/05/bear/

Offline Reinforcement Learning

Recap: Generalized Policy Iteration (GPI)

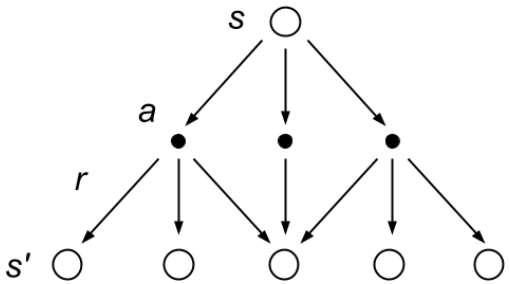




Offline Reinforcement Learning

Recap: Generalized Policy Iteration (GPI)

Policy Evaluation: For a given policy π, find Q



- The true state-action-value function satisfies the Bellman Equation:

$$\mathrm{Q}(s,a) = r(s,a) + \gamma \mathbb{E}[Q(s', \pi(s'))] = T_\pi Q(s,a)$$

- Approximate dynamic programming is used to minimize the Mean Squared Bellman Error:

$$\min_\theta \mathbb{E}_{s \sim d_\pi,\, a \sim \pi} \left[\left(Q_\theta(s,a) - T_\pi Q_\theta(s,a) \right)^2 \right]$$

Policy Improvement: For a given value function Q for π, find a better policy π_{new}

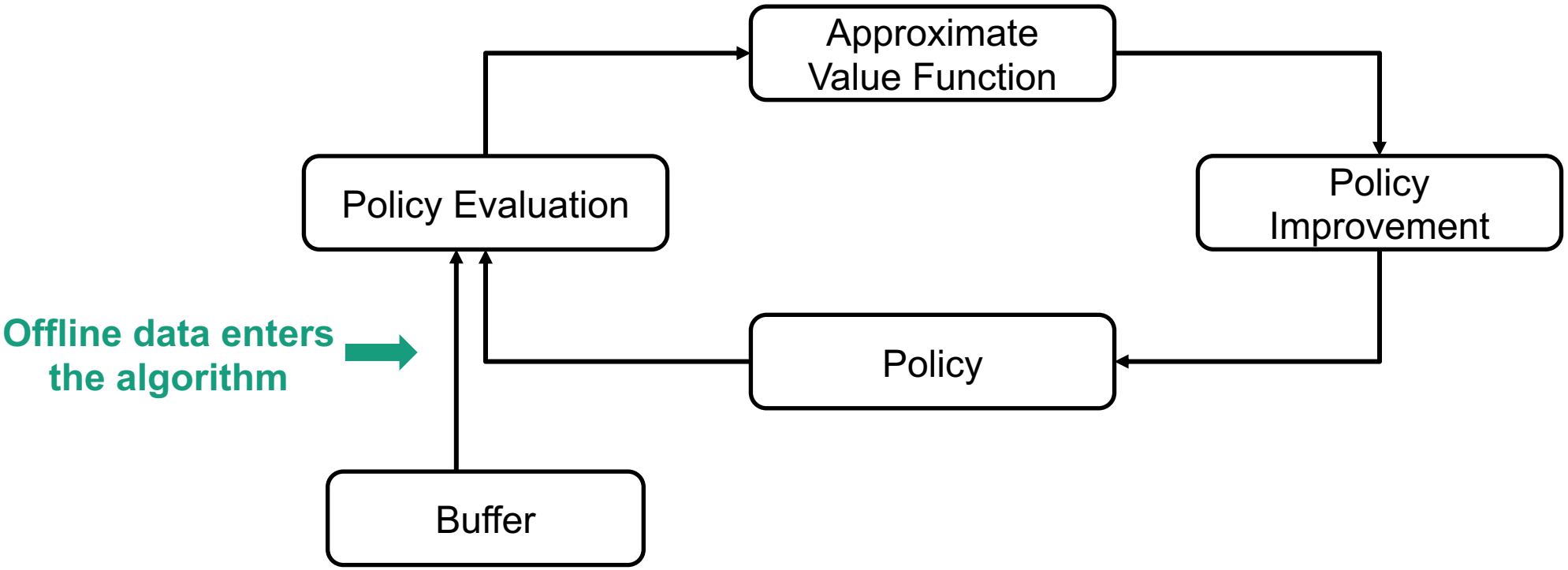
$$\pi_{new}(s) = \arg\max_a Q(s,a)$$

Policy Improvement Theorem:

- Let π and π' be two policies with $\forall\, s \in S$: $Q_\pi\big(s, \pi'(s)\big) \geq V_\pi(s)$, then $\forall\, s \in S$: $V_{\pi'}(s) \geq V_\pi(s)$

Offline Reinforcement Learning

GPI in the Offline Setting



Offline Reinforcement Learning

Offline Policy Evaluation and Distribution Shift

- What we want to do:

$$\min_{\theta} \mathbb{E}_{s\sim d_{\pi},\, a\sim\pi}\left[\left(r(s,a)+\gamma\mathbb{E}_{s'\sim p(s'|s,a),\, a'\sim\pi(s')}[Q_{\theta}(s',a')]-Q_{\theta}(s,a)\right)^{2}\right]$$

- What we would be doing naively:

$$\min_{\theta}\sum_{s\in S,a\in A}\left[\sum_{(s,a,r,s')\in\mathcal{D}_{|s,a}}\left(r+\gamma Q_{\theta}\left(s',\pi(s')\right)-Q_{\theta}(s,a)\right)\right]^{2}$$

- Which is the empirical approximation for:

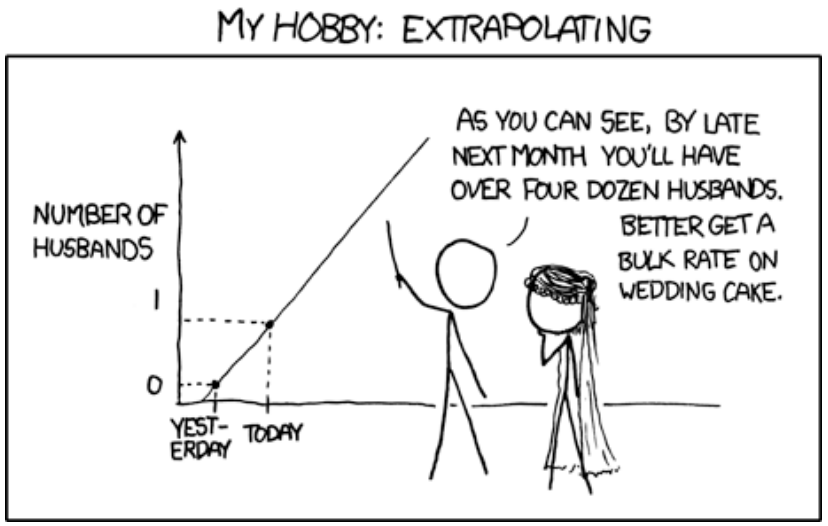
$$\min_{\theta} \mathbb{E}_{s\sim d_{\pi_{\beta}},\, a\sim\pi_{\beta}}\left[\left(r+\gamma\mathbb{E}_{s'\sim p(s'|s,a'),a'\sim\,\pi(s')}[Q_{\theta}(s',a')]-Q_{\theta}(s,a)\right)^{2}\right]$$

→**State and action distribution shift (the next action can be controlled)**

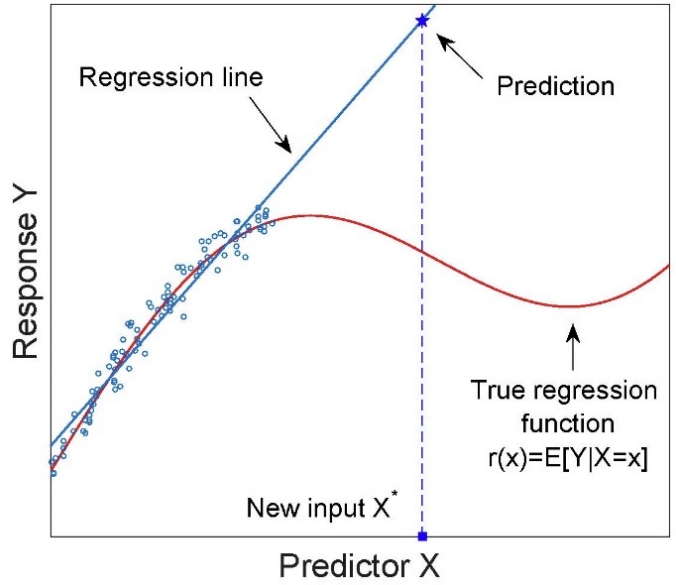
Offline Reinforcement Learning

Offline Policy Evaluation and Distribution Shift

- **The problem:** $P_{test} \neq P_{train}$
- Problematic in many domains:
  - How well does my model extrapolate
  - Adversarial examples
- In RL we have no ground-truth
  - bootstrapping the target makes the situation even worse



https://xkcd.com/605/



https://stats.stackexchange.com/questions/219579/what-is-wrong-with-extrapolation

Offline Reinforcement Learning

Offline Policy Evaluation and Distribution Shift

What we want to do:

$$\mathbb{E}_{s\sim d_\pi,\, a\sim\pi}\left[\left(r+\gamma\mathbb{E}_{s'\sim p(s'|s,a),\, a'\sim\pi(s')}[Q_\theta(s',a')]-Q_\theta(s,a)\right)^2\right]$$

What we would naively do:

$$\mathbb{E}_{s\sim d_{\pi_\beta},\, a\sim\pi_\beta}\left[\left(r+\gamma\mathbb{E}_{s'\sim p(s'|s,a),\, a'\sim\pi(s')}[Q_\theta(s',a')]-Q_\theta(s,a)\right)^2\right]$$

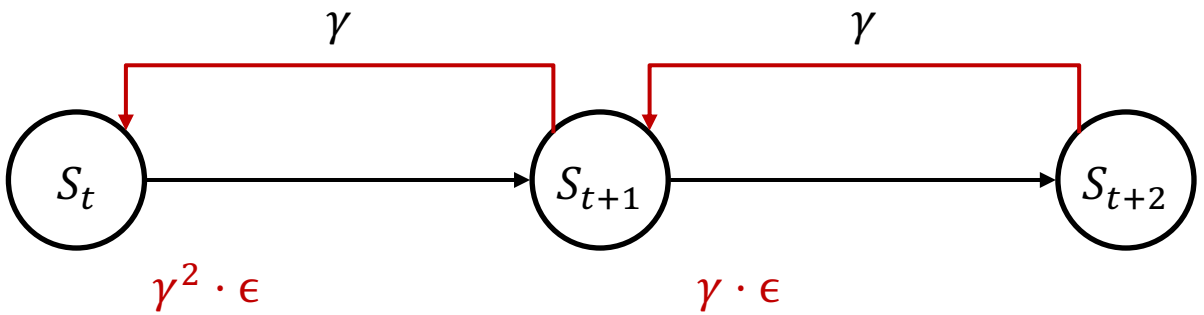
- **State distribution shift:**
  - Problem arises during **test time**
  - Does not invalidate the learned strategy on the states in $\mathcal{D}$ because unobserved states are never queried during training
- **Action distribution shift:**
  - Already problematic during **training** as inaccurate action values are used as bootstrapped targets
  - Can invalidate the learned strategy even on states in $\mathcal{D}$

Offline Reinforcement Learning

Bootstrapping Out of Distribution Actions

- We can view policy evaluation as a regression problem with a bootstrapped target: $y = r + \gamma Q_\theta(s', \pi(s'))$
- The objective is a (variant of) the mean squared error: $\left(y - Q_\theta(s,a)\right)^2$

- Due to distribution shift the bootstrapped y can be inaccurate as $\pi(s')$ might be an out of distribution action
- Errors are propagated through the state space and can potentially *pollute* everything



$\epsilon = Q_\theta(s,a) - Q(s,a)$

Offline Reinforcement Learning

Importance Sampling

- A classical method for off-policy training is **importance sampling**
- Define the **importance ratio**

$$\rho(s,a) = \frac{\pi(a|s) \times d_\pi(s)}{\pi_\beta(a|s) \times d_{\pi_\beta(s)}}$$

- Adjust the policy evaluation objective

$$\sum_{s \in S, a \in A} \rho(s,a) \left[\sum_{(s,a,r,s') \in \mathcal{D}_{|s,a}} r + \gamma Q_\theta(s', \pi(s')) - Q_\theta(s,a) \right]^2$$

- This has the actual loss in expectation

$$\mathbb{E}_{s \sim d_\pi,\, a \sim \pi} \left[\left(r + \gamma \mathbb{E}_{s' \sim p(s'|s,a), a' \sim \pi(s')} [Q_\theta(s', a')] - Q_\theta(s,a) \right)^2 \right]$$

Offline Reinforcement Learning

Importance Sampling

- The adjusted objective becomes

$$\sum_{s\in S, a\in A}\left(\sum_{(s,a,r,s')\in\mathcal{D}_{|s,a}}\frac{\pi(a|s)\times d_{\pi}(s)}{\pi_{\beta}(a|s)\times d_{\pi_{\beta}(s)}}(r+\gamma Q_{\theta}(s',\pi(s'))-Q_{\theta}(s,a))\right)^{2}$$

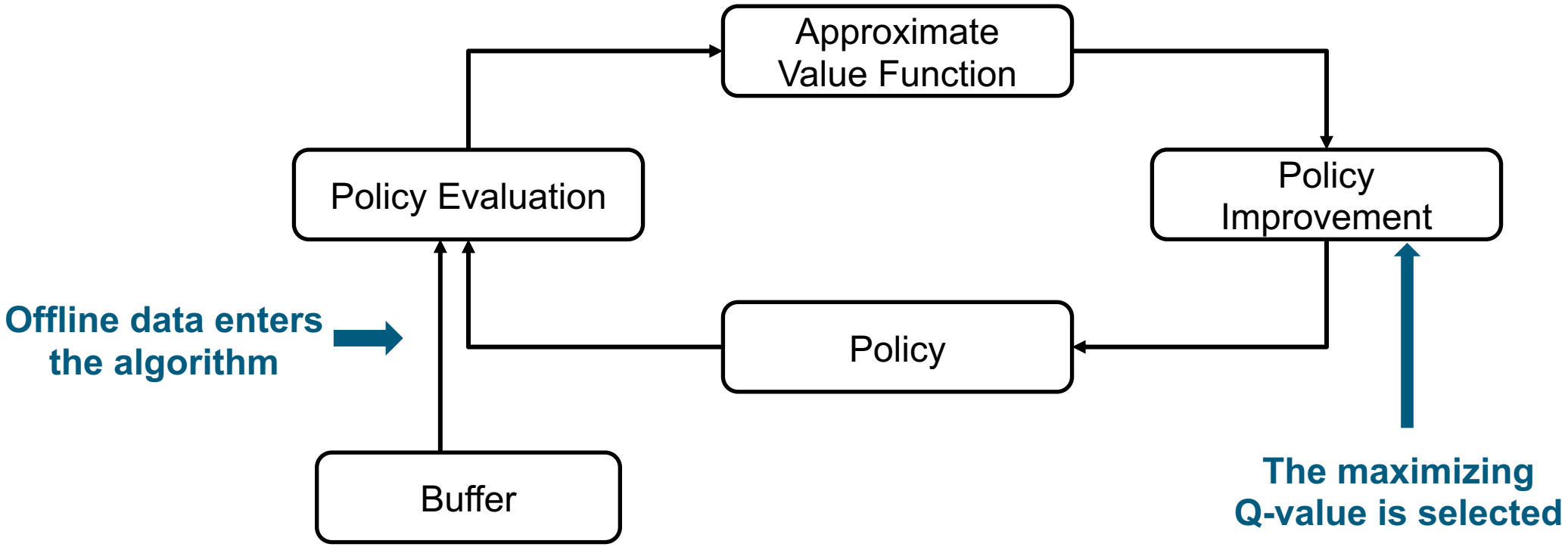
  - $a \in A$ with $\pi_{\beta}(a|s) \approx 0$ is an out of distribution action
  - Consider $\pi_{\beta}(a|s) = 0$ and $\pi(a|s) > 0$

Shortcomings

- Importance sampling induces high variance if the importance ratios are large
- The importance ratio is large if π and π_{β} differ strongly
- Density estimation in high-dimensional states is notoriously difficult
- There are methods that use importance sampling
  (see e.g. Off-Policy Policy Gradient with State Distribution Correction by Liu et al. (2019))
- In many cases we do not know π_{β}!

Offline Reinforcement Learning

Offline RL and Policy Improvement

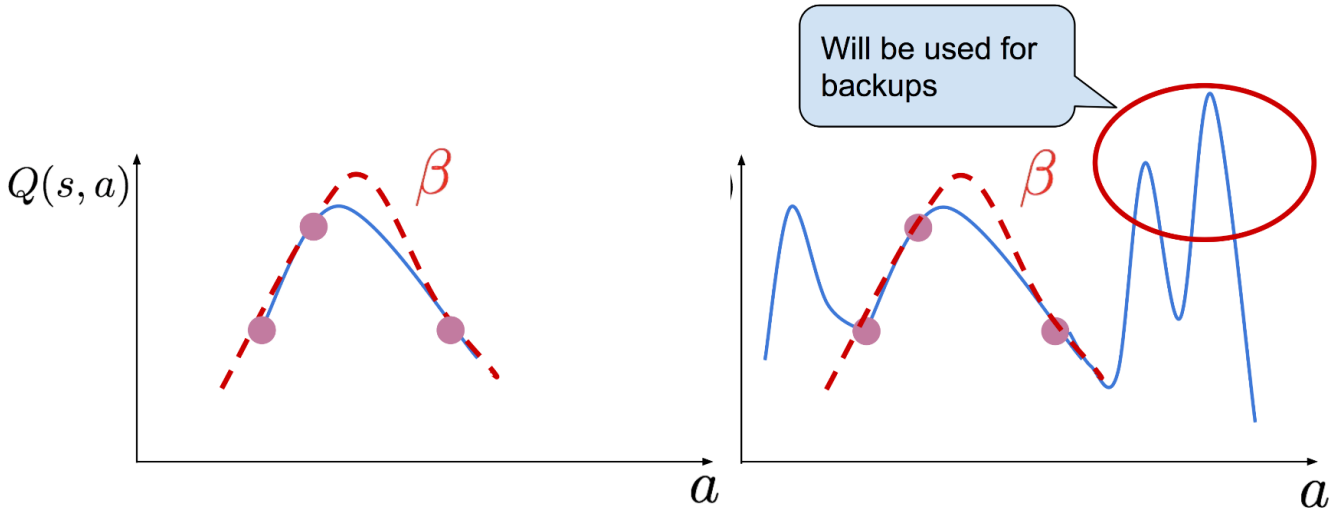


Offline Reinforcement Learning

Policy Improvement and Bootstrapping

- We can view policy evaluation as a regression problem with a bootstrapped target: $y = r + \gamma Q_\theta(s', \pi(s'))$
- The objective is a (variant of) the mean squared error: $\left(y - Q_\theta(s,a)\right)^2$

- The **maximizing** action is bootstrapped
  - This potentially draws the policy to out of distribution actions with large positive extrapolation error
  - The importance ratio $\rho(s,a)$ becomes worse



https://bair.berkeley.edu/blog/2019/12/05/bear/

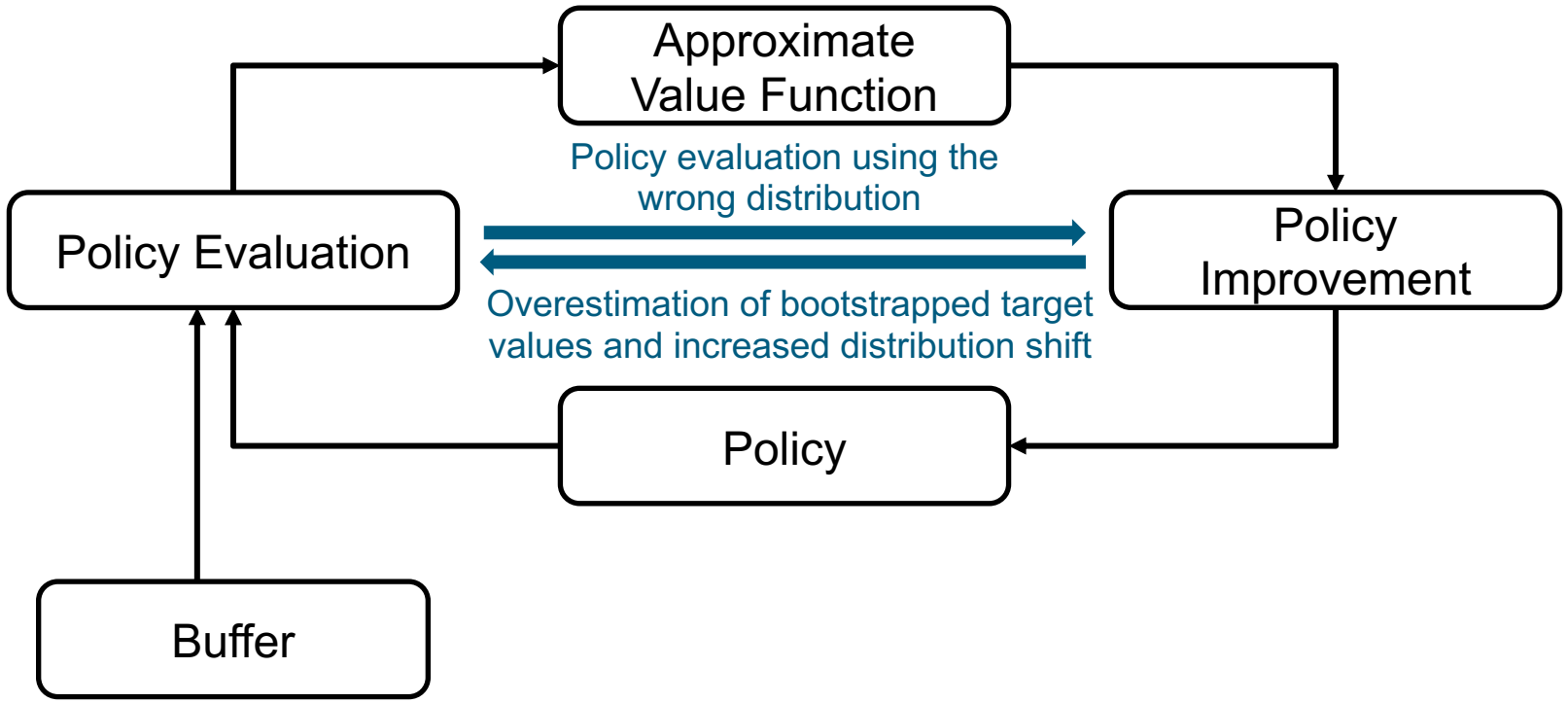
Offline Reinforcement Learning

Extrapolation Error in Active Reinforcement Learning

- In active reinforcement learning data is e.g. sampled using ϵ-greedy strategies:
  - Feedback for (potentially overestimated) actions is collected
  - Overestimated Q-values are going to be corrected downwards

Offline Reinforcement Learning

GPI in the Offline Setting



Offline Reinforcement Learning

The Dilemma

- We want to improve upon the behaviour policy
  → π should differ from π_β

- We want to be able to evaluate π using data from π_β
  → π should be similar to π_β

→ A tricky trade-off!

Offline Reinforcement Learning

The Deadly Triad

- The main challenges in Offline Reinforcement Learning arise because of the *Deadly Triad:*

- **Bootstrapping**
  - Needed for efficient learning

- **Function Approximation**
  - Needed for generalization

- **Distribution Shift**
  - A result of off-policy learning

Offline Reinforcement Learning

Agenda

- What is Offline Reinforcement Learning?
- Challenges of Offline Reinforcement Learning
- **Solution Approaches:**
  - Policy-constrained Methods:
    - Batch-constrained Q-Learning (BCQ)
    - Bootstrapping Error Accumulation Reduction (BEAR)
  - Conservative Methods:
    - Conservative Q-Learning (CQL)
    - Model-based Offline Policy Optimization (MOPO)
    - Conservative Offline Model-Based Policy Optimization (COMBO)
- Summary

Offline Reinforcement Learning

Solution Approaches

Policy-constrained Methods:

- Tackle the problem in the **policy improvement** step by keeping π close to π_β using distance $d(\pi, \pi_\beta)$
- What to use as d?
  - Match the distributions: Batch-constrained Q-Learning (BCQ)
  - Match the support: Bootstrapping Error Accumulation Reduction (BEAR)

Conservative Methods:

- Tackle the problem in the **policy evaluation** step by being conservative in areas of high uncertainty
  - Learn a lower bound for the true Q-Function: Conservative Q-Learning (CQL)
  - Learn a model with a pessimistic reward function: Model-based Offline Policy Optimization (MOPO)
  - Learn a lower bound for the true Q-Function and use additional data from a model: Conservative Offline Model-Based Policy Optimization (COMBO)

Offline Reinforcement Learning

Policy-Constrained Methods

- What is Offline Reinforcement Learning?
- Challenges of Offline Reinforcement Learning
- **Solution Approaches:**
  - **Policy-constrained Methods:**
    - Batch-constrained Q-Learning (BCQ)
    - Bootstrapping Error Accumulation Reduction (BEAR)
  - Conservative Methods:
    - Conservative Q-Learning (CQL)
    - Model-based Offline Policy Optimization (MOPO)
    - Conservative Offline Model-Based Policy Optimization (COMBO)
- Summary

Offline Reinforcement Learning

Policy-Constrained Methods

- The problems arise because the maximizing action is selected without uncertainty considerations

$$\pi_{new}(s) = \arg\max_a Q_\theta(s,a)$$

- Define the admissible set of policies $\Pi_\epsilon = \{\pi \mid d(\pi, \pi_\beta) \leq \epsilon)$ where d is a distance measure

- Consider a constrained policy improvement step

$$\pi_{new} = \arg\max_{\pi \in \Pi_\epsilon} \mathbb{E}[Q_\theta(s, \pi(s))]$$

Policy-Constrained Offline RL

Batch-Constrained Q-Learning (BCQ)

- **Batch-Constrained Q-Learning**[1] was originally introduced 2019
- General idea:
  - Restrict the action space in order to force the agent towards behaving close to on-policy with respect to the subset of the given data
  - Assumption: if bootstrapped actions are close, then the extrapolation error will be low

- The algorithm will be explained in three steps:
  1. Theoretical BCQ in the tabular case
  2. BCQ with function approximation in the discrete case
  3. BCQ with function approximation in the continuous case

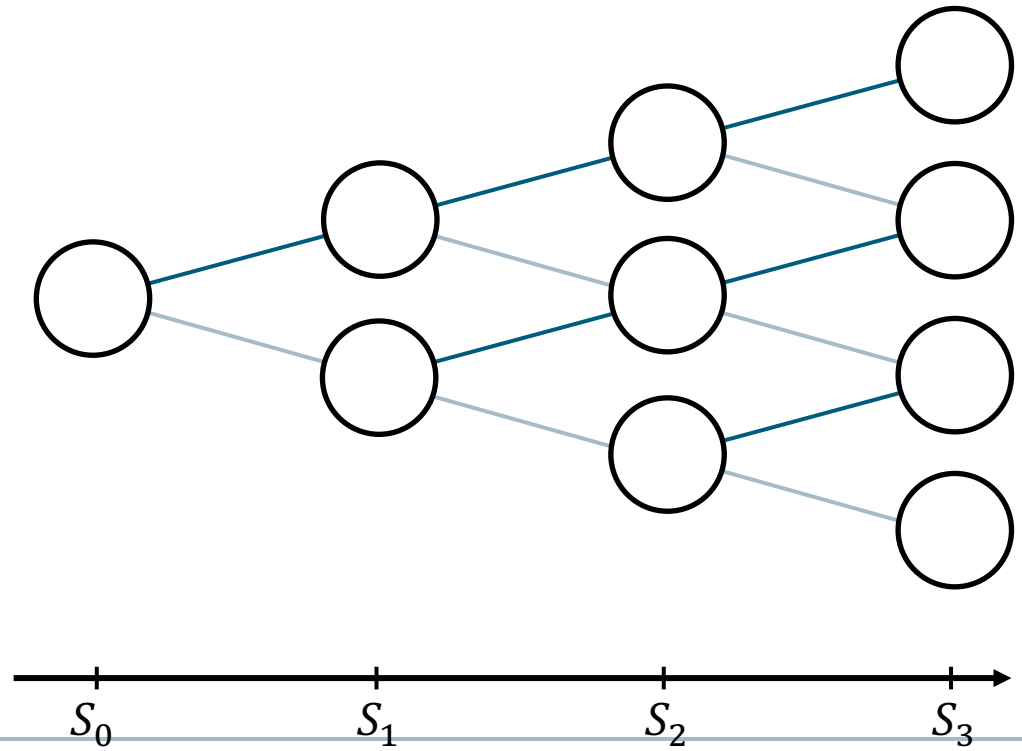
[1] *Fujimoto et a.: Off Policy Deep Reinforcement Learning without Exploration. ICML. 2019.*

Policy-Constrained Offline RL

BCQ – Deterministic Tabular Case

Finite, Deterministic MDP

- Actions: up, down
- Tabular Q-Function
- Rewards are neglected for simplicity



Fraunhofer IIS

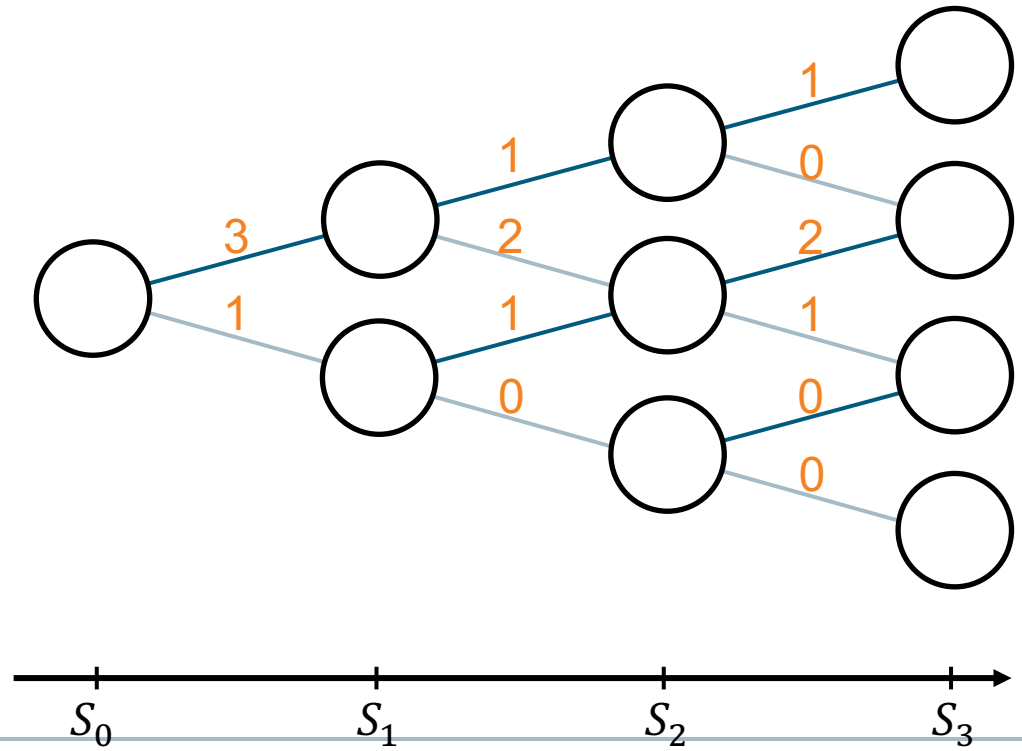
Policy-Constrained Offline RL

BCQ – Deterministic Tabular Case

Finite, Deterministic MDP

- Actions: up, down
- Tabular Q-Function
- Rewards are neglected for simplicity

- Orange numbers are the observed transition counts t_i
- $\mathcal{D}$ consists of the transitions, where $|t_i| > 0$



Fraunhofer IIS

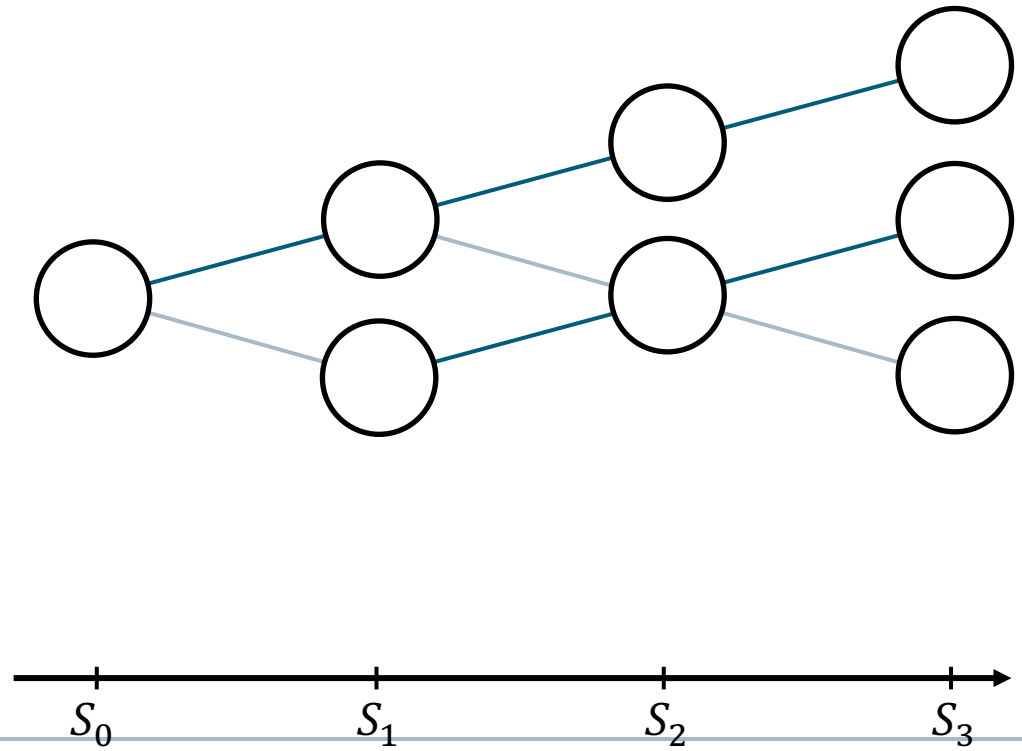
Policy-Constrained Offline RL

BCQ – Deterministic Tabular Case

Finite, Deterministic MDP

- Actions: up, down
- Tabular Q-Function
- Rewards are neglected for simplicity

- Estimate the restricted MDP
- Solve the restricted MDP (e.g., using Q-Learning)
- All policies that stay lie in the restricted MDP can be accurately assessed because of the determinism

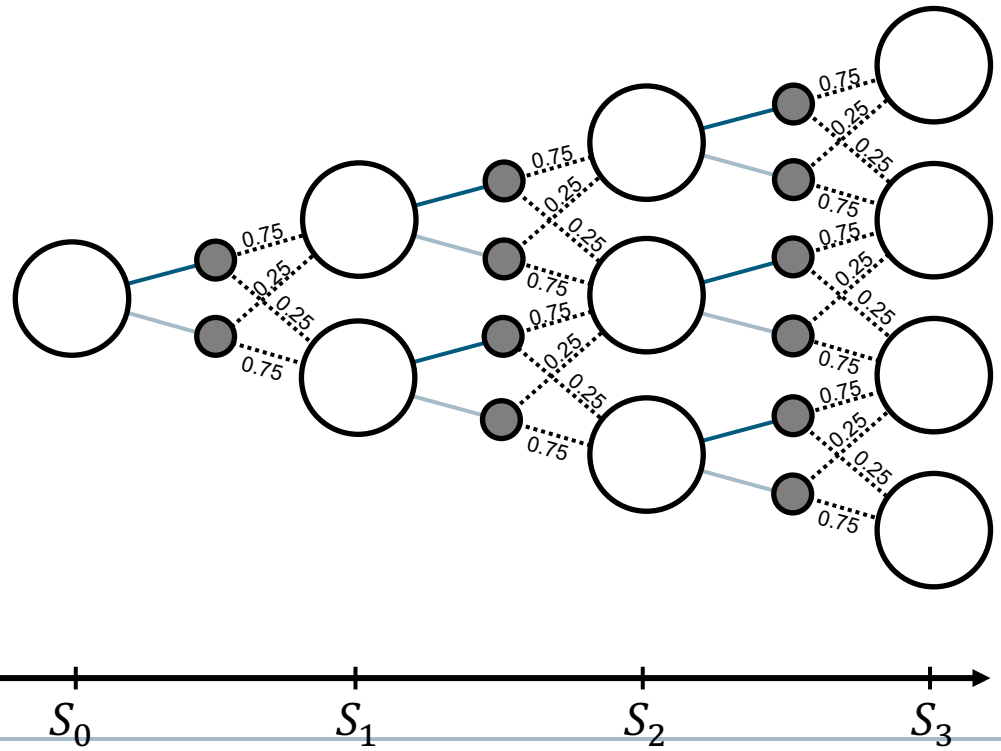


Policy-Constrained Offline RL

BCQ – Stochastic Tabular Case

Finite, Stochastic MDP

- Actions: up, down
- Tabular Q-Function
- Rewards are neglected for simplicity



Fraunhofer IIS

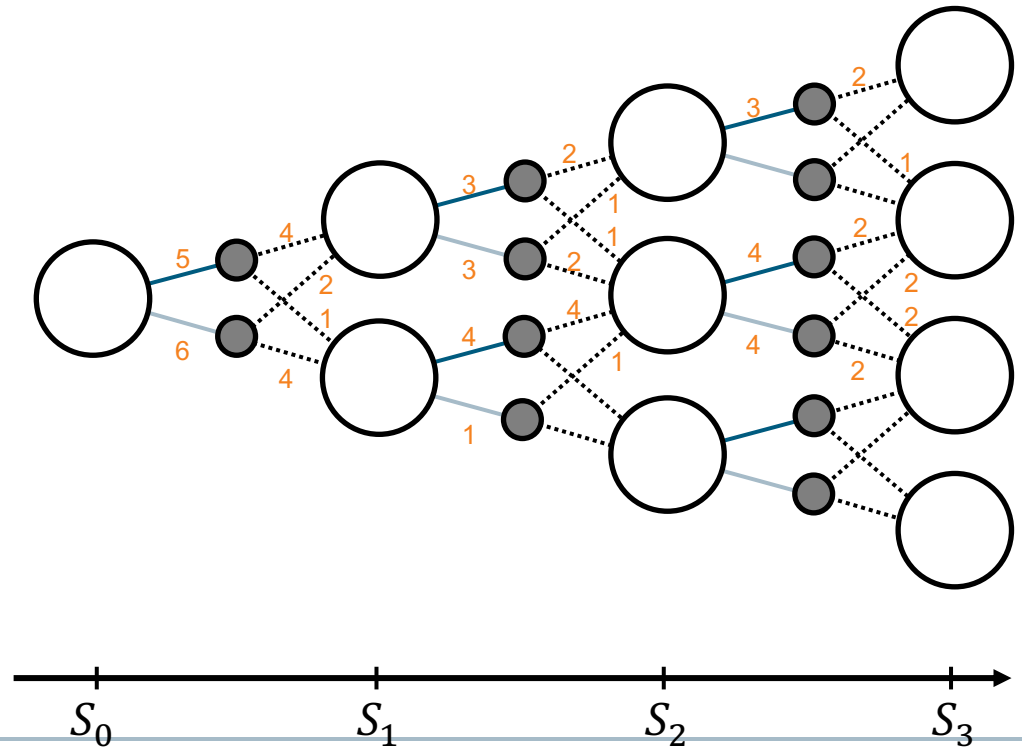
Policy-Constrained Offline RL

BCQ – Stochastic Tabular Case

Finite, Stochastic MDP

- Actions: up, down
- Tabular Q-Function
- Rewards are neglected for simplicity

- Orange numbers are the observed transition counts t_i
- $\mathcal{D}$ consists of the transitions, where $|t_i| > 0$



Fraunhofer IIS

Policy-Constrained Offline RL

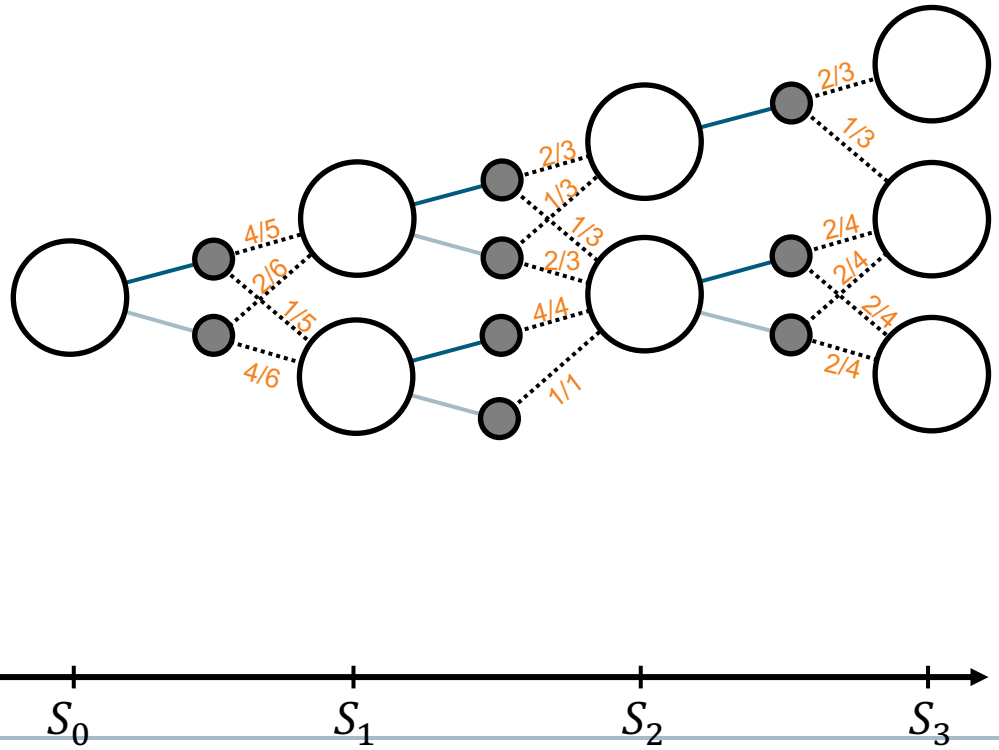
BCQ – Stochastic Tabular Case

Finite, Stochastic MDP

- Actions: up, down
- Tabular Q-Function
- Rewards are neglected for simplicity

- Estimate the restricted MDP
- Solve the estimated MDP (e.g., using Q-Learning)
- Quality of the learned policy depends on the quality of the estimation

- **When deploying the policy, the agent might be swept into the unknown!**



Fraunhofer IIS

Policy-Constrained Offline RL

BCQ with Function Approximation – Discrete Case

- Q-Learning:

$$\min_{\theta}(r + \gamma \max_{a' \in A(s')} Q_\theta(s', a') - Q_\theta(s, a))^2$$

- Let us define

$$A_\epsilon^{BCQ}(s) = \left\{ a \in A(s): \frac{\hat{\pi}_\beta(a|s)}{\max_a \hat{\pi}_\beta(a|s)} \geq \epsilon \right\},$$

  where $\epsilon \in [0,1]$ is the threshold parameter and $\hat{\pi}_\beta$ an estimate for the behaviour policy

- The **constrained target** is $y = r + \gamma \cdot \max_{a' \in A_\epsilon^{BCQ}(s')} Q\,(s', a')$
  - $\epsilon = 1$ → behavioural cloning
  - $\epsilon = 0$ → Q-Learning

- The learned policy is $\pi(s) = \arg\max_{a \in A_\epsilon^{BCQ}(s')} Q(s, a)$

Policy-Constrained Offline RL

BCQ with Function Approximation – Continuous Case

- **Problem:**
  - Computing the restriction $\frac{\hat{\pi}_\beta(a|s)}{\max_a \hat{\pi}_\beta(a|s)} > \epsilon$ is not so straightforward in the continuous case

- **BCQ addresses this using three counter-measures:**
  - A **Conditional Variational Autoencoder** $\hat{\pi}_\beta(\cdot\,|s)$ is used to sample actions
  - Additionally, a **perturbation model** improves the sampled actions $a_i \sim \hat{\pi}_\beta(\cdot\,|s)$ in a neighbourhood
  - **(Soft) Clipped Double Q-Learning**[1] fights overestimation in continuous action spaces

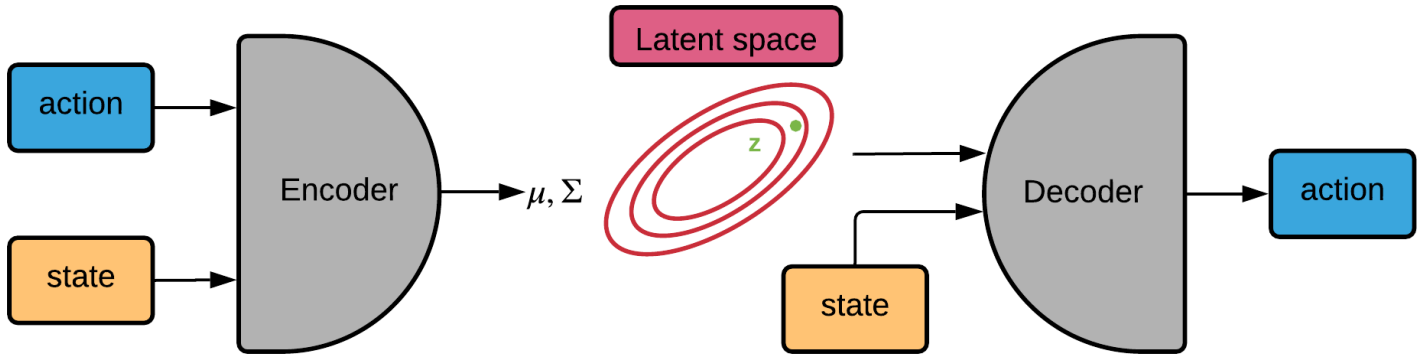
[1] *Fujimoto et al.: Addressing Function Approximation Error in Actor-Critic Methods. ICML. 2018.*

Policy-Constrained Offline RL

BCQ – Sampling from the Behaviour Policy

- **Goal:** Sample from the behavioural policy $\pi_\beta(\cdot\,|s)$
- **Problem:** $\pi_\beta(\cdot\,|s)$ is unknown and difficult to sample from
- **Solution:** **Conditional Variational Autoencoder**

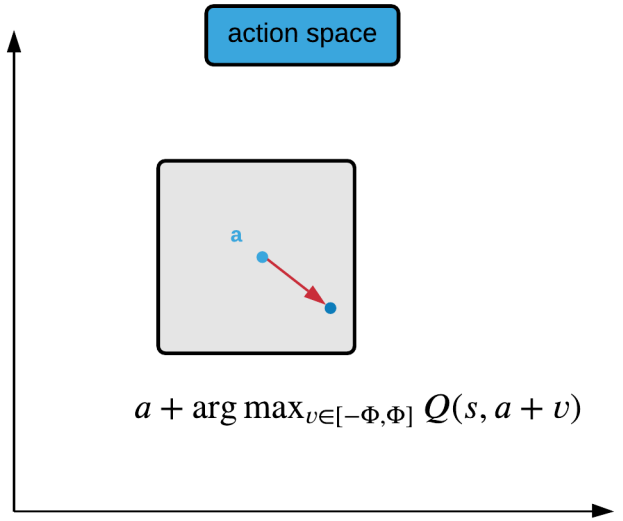
- Generate samples from a trained VAE:
  1. Sample z_i from the latent space (encoder is <u>not</u> used)
  2. Decode $a_i' = dec(z_i, s)$



Policy-Constrained Offline RL

BCQ – Perturbation Model

- **Situation:** In state s' we have obtained sampled actions a_i' from $\hat{\pi}_\beta(\cdot\,|s')$
- **Problem:** Finding a good action requires many samples
- **Solution:**
  - Train an additional perturbation model ξ that improves the Q-value of a_i in a neighbourhood
  - The goal is $\xi_\phi(s, a, \Phi) = \arg\max_{v \in [-\Phi, \Phi]} Q(s, a + v)$
    - $\Phi = 0$ → Behavioural cloning
    - $\Phi = \infty$ → Q-Learning
  - It is trained with the DDPG algorithm



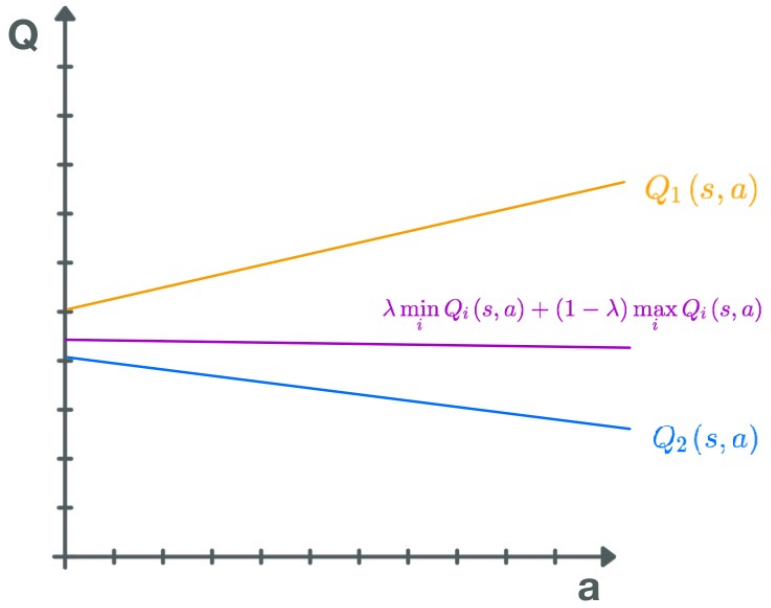
Policy-Constrained Offline RL

BCQ – Soft Clipped Double Q-Learning

- **Situation:** In state s we have perturbed actions a_i' from $\hat{\pi}_\beta(\cdot\,|s)$ resulting in $a_i^* = \xi(s, a_i, \Phi)$
- **Problem:** Vanilla Double Q-Learning can be ineffective when the networks are too similar
  → Overestimation of Q-values
- **Solution:**
  - Use a convex combination of the minimum and the maximum

$$y = \max_{a'=a_1^*,\ldots,a_k^*} \left\{ \lambda \cdot \min_i Q_{Q_{\theta_i'}}(s', a') + (1-\lambda) \cdot \max_i Q_{\theta_i'}(s', a') \right\}$$

  - When $\lambda \in [0,1]$ is large this penalizes **uncertain** regions



Policy-Constrained Offline RL

Continuous BCQ – Algorithm

Algorithm 1 BCQ

Input: Batch $\mathcal{B}$, horizon T, target network update rate τ, mini-batch size N, max perturbation Φ, number of sampled actions n, minimum weighting λ.

Initialize Q-networks $Q_{\theta_1}, Q_{\theta_2}$, perturbation network ξ_ϕ, and VAE $G_\omega = \{E_{\omega_1}, D_{\omega_2}\}$, with random parameters θ_1, θ_2, ϕ, ω, and target networks $Q_{\theta'_1}, Q_{\theta'_2}$, $\xi_{\phi'}$ with $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \phi' \leftarrow \phi$.

for $t = 1$ **to** T **do**

Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$

$\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$
$\omega \leftarrow \text{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\text{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$

Sample n actions: $\{a_i \sim G_\omega(s')\}_{i=1}^n$
Perturb each action: $\{a_i = a_i + \xi_\phi(s', a_i, \Phi)\}_{i-1}^n$

Set value target y (Eqn. 13)

$\theta \leftarrow \text{argmin}_\theta \sum (y - Q_\theta(s, a))^2$

$\phi \leftarrow \text{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$

Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$
$\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

end for

train CVAE

sample from CVAE and perturb

clipped target

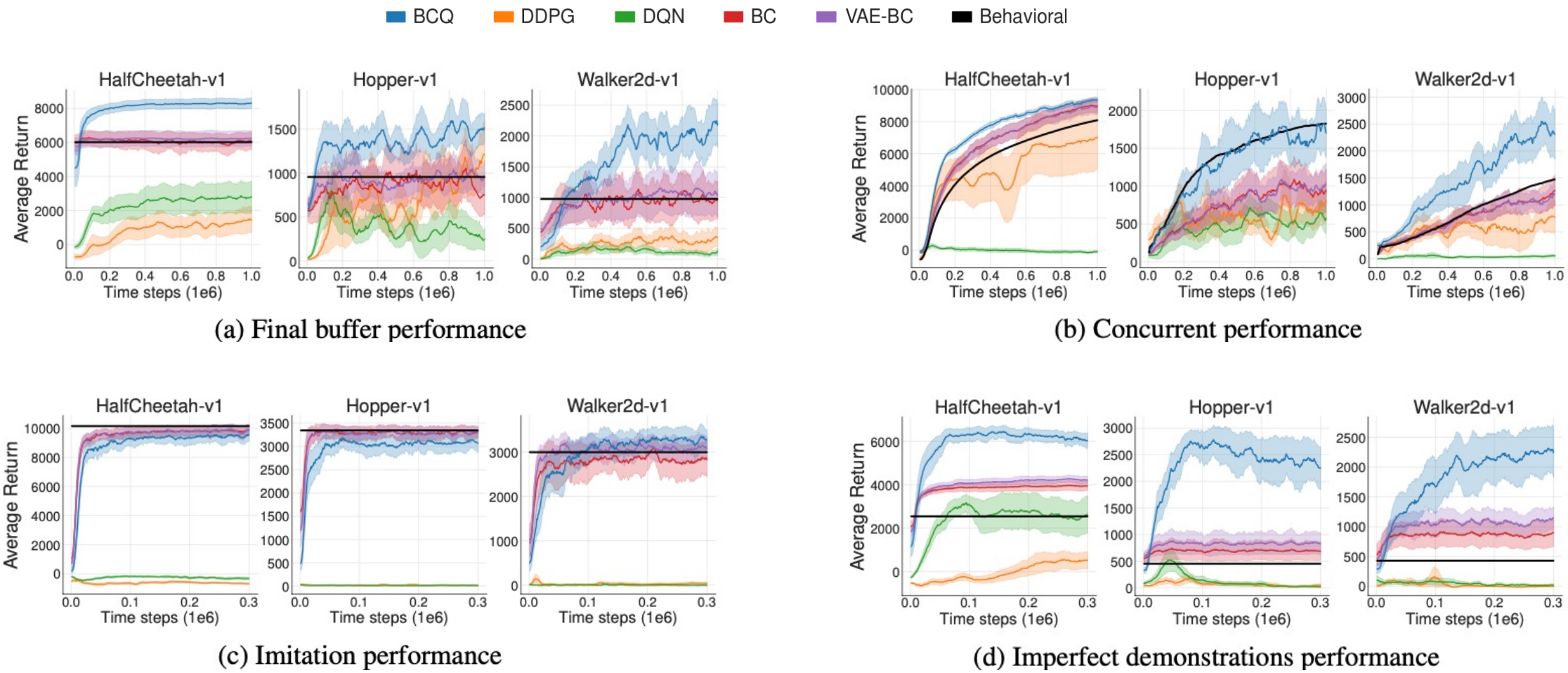
train Q

train perturbation

update targets

Policy-Constrained Offline RL

Continuous BCQ – Results



Fraunhofer IIS

Offline Reinforcement Learning

Policy-Constrained Methods

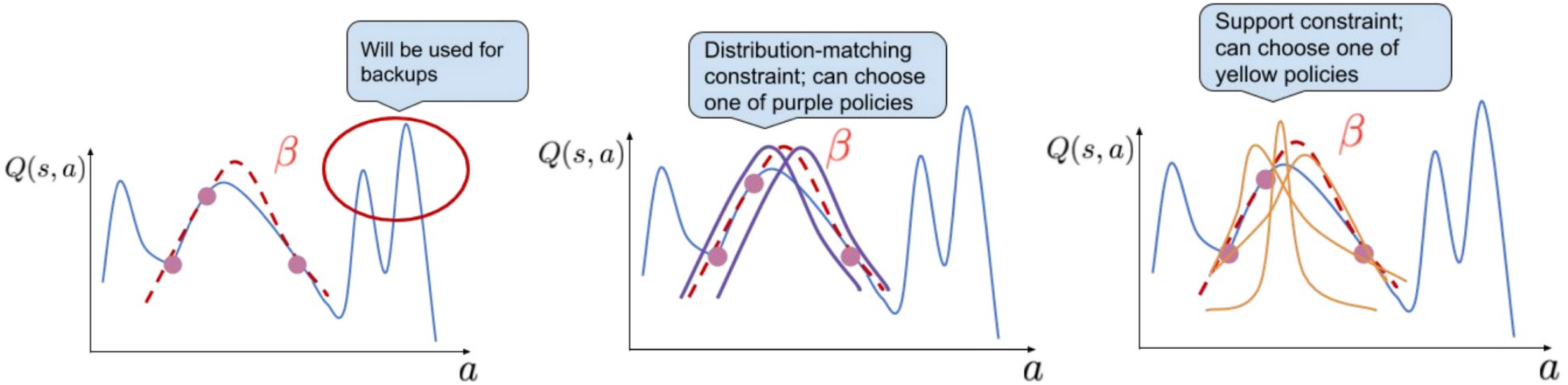
- What is Offline Reinforcement Learning?
- Challenges of Offline Reinforcement Learning
- Solution Approaches:
  - **Policy Constrained Methods:**
    - Batch Constrained Q-Learning (BCQ)
    - **Bootstrapping Error Accumulation Reduction (BEAR)**
  - Conservative Methods:
    - Conservative Q-Learning (CQL)
    - Model-based Offline Policy Optimization (MOPO)
    - Conservative Offline Model-Based Policy Optimization (COMBO)
- Summary

Policy-Constrained Offline RL

Bootstrapping Error Accumulation Reduction (BEAR)

General motivation of BEAR[1]:

- The constraint in BCQ is overly restrictive:
  If the behaviour policy is uniform, the learned policy must also be close to uniform!
- Only require that π lies in the support of π_β: $\pi(a|s) > 0 \Rightarrow \pi_\beta(a|s) > \epsilon$



https://bair.berkeley.edu/blog/2019/12/05/bear/

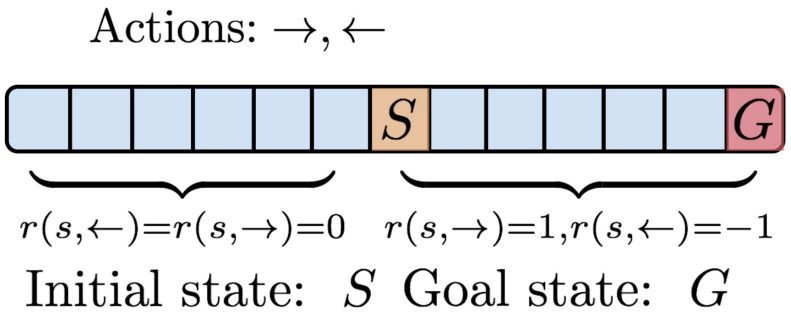
[1] *Kumar et al.: Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. NeurIPS 2019.*

Policy-Constrained Offline RL

Bootstrapping Error Accumulation Reduction (BEAR)

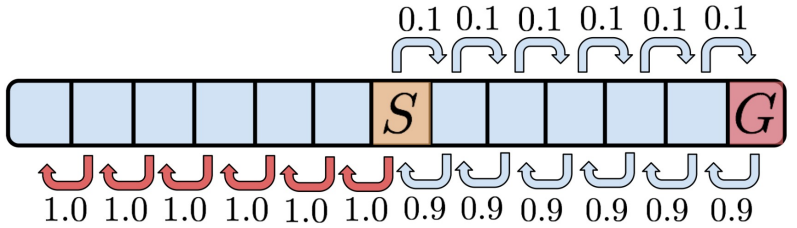
Example:

- 1-dimensional grid-world, two actions: left & right
- The agent starts in **S** and its goal is to reach **G**



1D-Lineworld Environment

- The behavior policy is used to generate our dataset
  - Right side: sub-optimal actions are more likely (90%), *but both actions are in-distribution at all the states*
  - Left side: only the left action is used



Behavior Policy

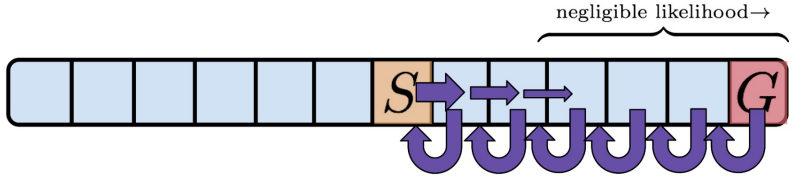
https://bair.berkeley.edu/blog/2019/12/05/bear/

Policy-Constrained Offline RL

Bootstrapping Error Accumulation Reduction (BEAR)

Example:

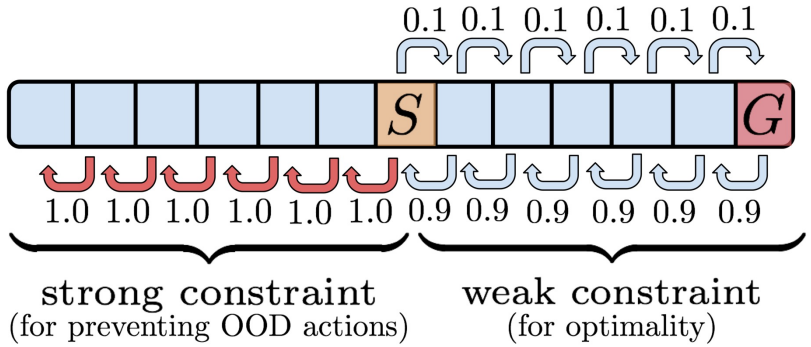
- **Distribution-matching** constraint can be arbitrarily sub-optimal
- Chances of reaching the goal become very small
- With bigger grid-worlds the chances approach 0



Learned Policy via distribution-matching

Why does distribution-matching fail here?

- Assume we use a penalty for distribution-matching
- If penalty tight: agent goes **left** between **S** and **G**
  → suboptimal behavior
- If penalty is generous (in hope for getting a better policy): agent will start to take OOD actions to left of **S**
  → affects Q-value of **S**
  → maybe the policy will even go left when being in **S**!



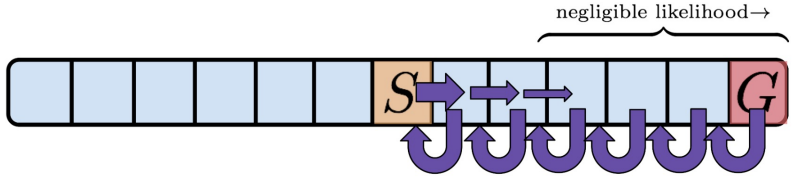
https://bair.berkeley.edu/blog/2019/12/05/bear/

Policy-Constrained Offline RL

Bootstrapping Error Accumulation Reduction (BEAR)

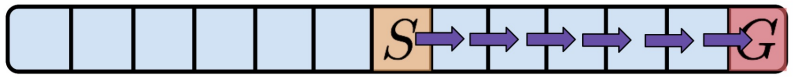
Example:

- **Distribution-matching** constraint can be arbitrarily sub-optimal
- Chances of reaching the goal become very small
- With bigger grid-worlds the chances approach 0



Learned Policy via distribution-matching

- In contrast, a **support-constraint** can recover the optimal policy with probability 1!



Learned Policy via support-constraint

https://bair.berkeley.edu/blog/2019/12/05/bear/

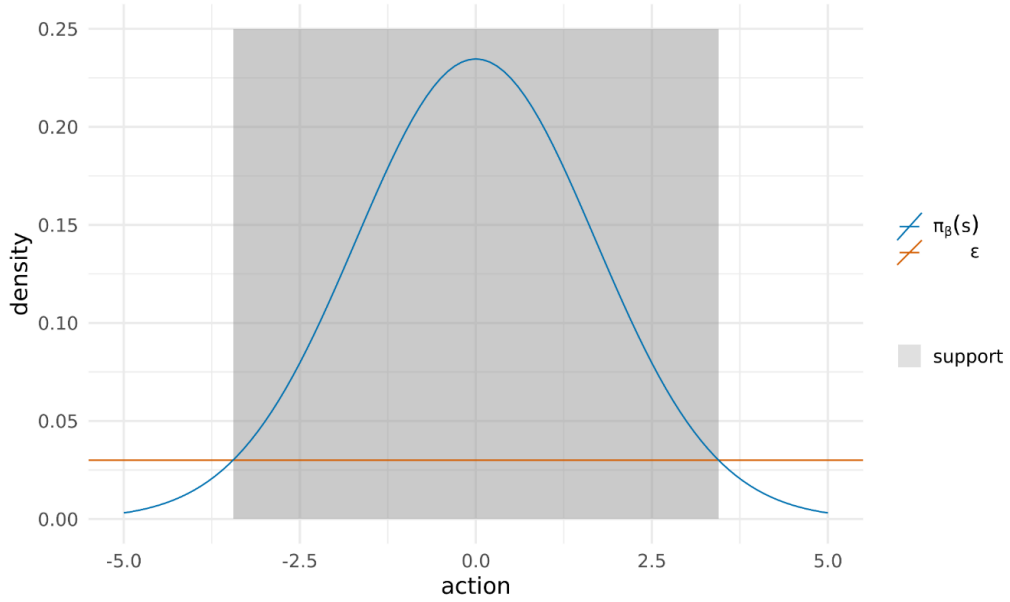
Policy-Constrained Offline RL

Bootstrapping Error Accumulation Reduction (BEAR)

- **Q-Learning**
  - Objective: $\min_{\theta}(r + \gamma \max_{a' \in A(s')} Q_\theta(s', a') - Q_\theta(s, a))^2$
  - Policy: $\pi(s) = \text{argmax}_{a \in A(s)} Q_\theta(s, a)$

Remember: $A_\epsilon^{BCQ}(s) = \left\{a \in A(s): \frac{\hat{\pi}_\beta(a|s)}{\max_a \hat{\pi}_\beta(a|s)} \geq \epsilon\right\}$,

- **BEAR**
  - Policy constraint: $A_\epsilon^{BEAR}(s) = \{a \in A(s): \hat{\pi}_\beta(a|s) \geq \epsilon\}$
  - Objective: $\left(r + \gamma \cdot \max_{a' \in A_\epsilon^{BEAR}(s')} Q_\theta(s', a') - Q_\theta(s, a)\right)^2$
  - Policy: $\pi(s) = \text{argmax}_{a \in A_\epsilon^{BEAR}(s)} Q_\theta(s, a)$



Fraunhofer IIS

Policy-Constrained Offline RL

BEAR – Support Constraint

- **Goal:**
  - get $\max_{a' \in A_\epsilon^{BEAR}(s')} Q(s', a')$ in a continuous action space
- **Approach:**
  - Train an actor π_ϕ that satisfies $\pi_\phi(a|s) > 0 \Rightarrow \pi_\beta(a|s) \geq \epsilon$
  - *Optimization problem:*

$$\max_\phi Q\big(s, \pi_\phi(s)\big) \quad s.t.\ \pi_\phi(a|s) > 0 \Rightarrow \pi_\beta(a|s) > \epsilon$$

  → This requires a distance metric $d(\pi, \pi_\beta)$ that measures the violation of the support constraint

- **Solution in BEAR:**
  - This constraint is implemented via **Maximum Mean Discrepancy** $\mathrm{MMD}(\pi_\phi, \pi_\beta)$
  - This turns the policy improvement step into a *constrained optimization problem:*

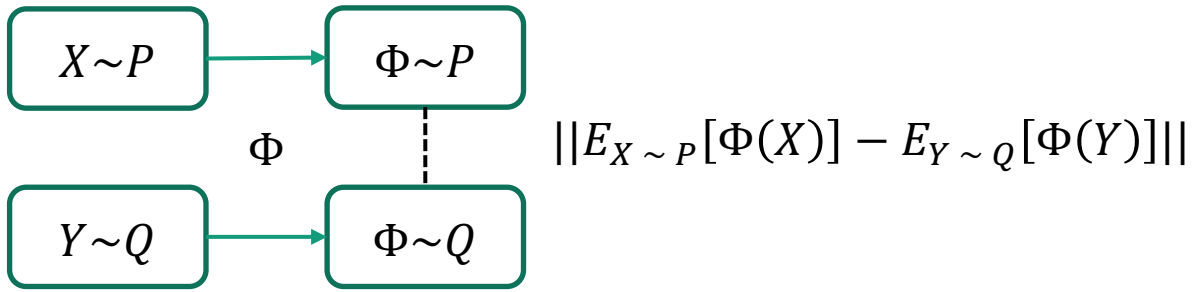
$$\max_\phi Q_\theta\left(s, \pi_\phi(s)\right) \quad s.t.\ \mathrm{MMD}^2\left(\pi_\phi, \pi_\beta\right) < C(\epsilon)$$

Policy-Constrained Offline RL

Parenthesis: Maximum Mean Discrepancy

- **Goal:** Measure the distance between distributions Q and P
  - Let $X \sim P$ and $Y \sim Q$
- **Idea of MMD:**
  1. Map X and Y into a Hilbert space space $(H, \langle \cdot \rangle)$ using a feature map Φ
  2. Calculate difference between their expectations in H:

$$||E_{X \sim P}[\Phi(X)] - E_{Y \sim Q}[\Phi(Y)]||$$



Policy-Constrained Offline RL

Parenthesis: Maximum Mean Discrepancy

- **Goal:** Measure the distance between distributions Q and P
  - Let $X \sim P$ and $Y \sim Q$
- **Kernel trick:** For certain feature maps Φ it holds that $\langle \Phi(x), \Phi(y) \rangle = k(x, y)$
  - Here k is a RBF kernel (e.g. the Gaussian kernel)
  - In these cases it holds that[1]

$$||\mathbb{E}_{X \sim P}[\Phi(X)] - \mathbb{E}_{Y \sim Q}[\Phi(Y)]|| = \mathbb{E}_{X,X' \sim P}[k(X, X')] + \mathbb{E}_{Y,Y' \sim Q}[k(Y, Y')] + 2\mathbb{E}_{X \sim P, Y \sim Q}[k(X, Y)]$$

[1] *Tolstikhin et al.: Minimax Estimation of Maximum Mean Discrepancy with Radial Kernels. NIPS. 2016.*

Policy-Constrained Offline RL

Parenthesis: Maximum Mean Discrepancy

only for reference

- **Goal:** Measure the distance between distributions Q and P
  - Let $X \sim P$ and $Y \sim Q$
  - Estimate $\mathrm{MMD}^2(\pi_\phi, \pi_\beta)$ from samples $a_1^\phi, \dots, a_n^\phi \sim \pi_\phi\,(s)$ and $a_1^\beta, \dots, a_m^\beta \sim \hat{\pi}_\beta(s)$:

$$\frac{1}{n^2}\sum_{i,i'} k\left(a_i^\phi, a_{i'}^\phi\right) - \frac{2}{nm}\sum_{i,j} k\left(a_i^\phi, a_j^\beta\right) + \frac{1}{m^2}\sum_{j,j'} k\left(a_j^\beta, a_{j'}^\beta\right)$$

  - For k they use the Gaussian kernel:

```
def gaussian_kernel(x, y, sigma=0.1):
  return exp(-(x - y).pow(2).sum() / (2 * sigma.pow(2)))

def compute_mmd(x, y):
  k_x_x = gaussian_kernel(x, x)
  k_x_y = gaussian_kernel(x, y)
  k_y_y = gaussian_kernel(y, y)
  return sqrt(k_x_x.mean() + k_y_y.mean() - 2*k_x_y.mean())
```

  - Empirically, this sample-estimated distance matches the support in the low-intermediate sample regime[1]

https://bair.berkeley.edu/blog/2019/12/05/bear/

[1] Kumar et al.: Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. NeurIPS 2019.

Policy-Constrained Offline RL

Dual Gradient Descent

- Objective for the actor:

$$\max_{\phi} Q_\theta\left(s, \phi(s)\right) \qquad s.t.\ MMD^2\left(A^\phi, A^\beta\right) \leq \epsilon$$

  - This is a **constrained optimization** problem
  - Many algorithms exist to solve this, e.g., penalty and barrier methods
  - BEAR uses **Dual Gradient Descent**

Policy-Constrained Offline RL

Dual Gradient Descent in a Nutshell

only for reference

- Optimization problem (for equality constraints):

$$\max_x \; f(x) \quad s.t.\; C(x) = 0$$

- Lagrangian:

$$L(x, \lambda) = f(x) + \lambda C(x)$$

- The dual gradient descent algorithm solves the **dual problem** instead of the original (primal) problem
  - The dual problem is: $\max_x \min_\lambda L(x, \lambda)$
  - Define $g(\lambda) = L(x^*(\lambda), \lambda)$, where $x^*(\lambda) = \arg\max_x L(x, \lambda)$
  - **Algorithm**
    1. Find $x^* \leftarrow \arg\max_x L(x, \lambda)$
    2. Compute $\frac{dg}{d\lambda} = \frac{dL}{d\lambda}(x^*, \lambda)$
    3. $\lambda \leftarrow \lambda + \beta \frac{dg}{d\lambda}$

see also https://jonathan-hui.medium.com/rl-dual-gradient-descent-5e524c1f049

Policy-Constrained Offline RL

BEAR – Algorithm

only for reference

Algorithm 1 BEAR Q-Learning (BEAR-QL)

input : Dataset $\mathcal{D}$, target network update rate τ, mini-batch size N, sampled actions for MMD n, minimum λ

1: Initialize Q-ensemble $\{Q_{\theta_i}\}_{i=1}^{K}$, actor π_ϕ, Lagrange multiplier α, target networks $\{Q_{\theta'_i}\}_{i=1}^{K}$, and a target actor $\pi_{\phi'}$, with $\phi' \leftarrow \phi, \theta'_i \leftarrow \theta_i$

2: **for** t in $\{1, \ldots, \mathrm{N}\}$ **do**

3: Sample mini-batch of transitions $(s, a, r, s') \sim \mathcal{D}$

Q-update:

4: Sample p action samples, $\{a_i \sim \pi_{\phi'}(\cdot|s')\}_{i=1}^{p}$

5: Define $y(s,a) := \max_{a_i}[\lambda \min_{j=1,..,K} Q_{\theta'_j}(s', a_i) + (1-\lambda)\max_{j=1,..,K} Q_{\theta'_j}(s', a_i)]$

6: $\forall i, \theta_i \leftarrow \arg\min_{\theta_i}(Q_{\theta_i}(s,a) - (r + \gamma y(s,a)))^2$

Policy-update:

7: Sample actions $\{\hat{a}_i \sim \pi_\phi(\cdot|s)\}_{i=1}^{m}$ and $\{a_j \sim \mathcal{D}(s)\}_{j=1}^{n}$, n preferably an intermediate integer(1-10)

8: Update ϕ, α by minimizing Equation 1 by using dual gradient descent with Lagrange multiplier α

9: **Update Target Networks:** $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$; $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$

10: **end for**

$$\pi_\phi := \max_{\pi \in \Delta_{|S|}} \mathbb{E}_{s\sim\mathcal{D}} \mathbb{E}_{a\sim\pi(\cdot|s)} \left[\min_{j=1,..,K} \hat{Q}_j(s,a) \right] \quad \text{s.t. } \mathbb{E}_{s\sim\mathcal{D}}[\mathrm{MMD}(\mathcal{D}(s), \pi(\cdot|s))] \leq \varepsilon \qquad (1)$$

Policy-Constrained Offline RL

BEAR – Results

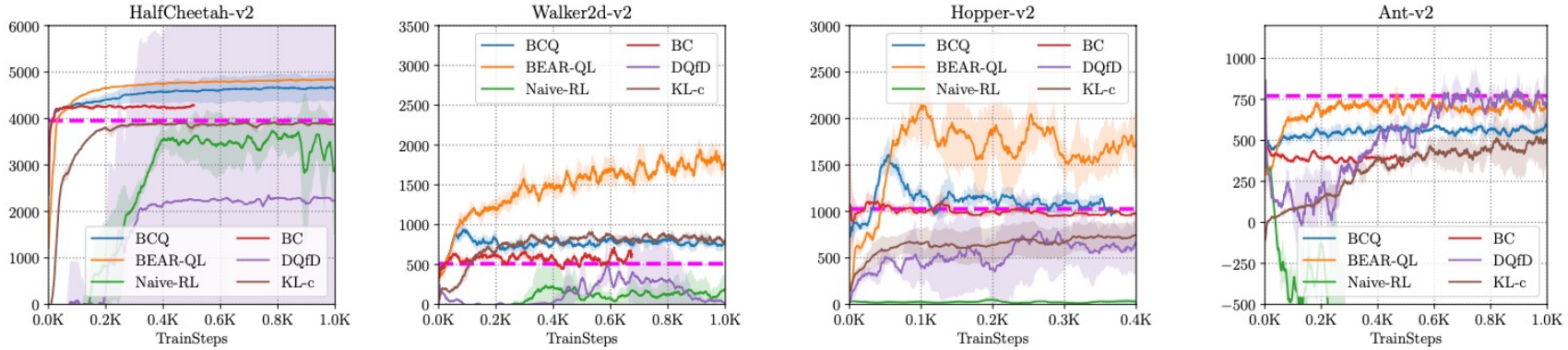


Figure 3: Average performance of BEAR-QL, BCQ, Naïve RL and BC on medium-quality data averaged over 5 seeds. BEAR-QL outperforms both BCQ and Naïve RL. Average return over the training data is indicated by the magenta line. One step on the x-axis corresponds to 1000 gradient steps.

Policy-Constrained Offline RL

BEAR – Results

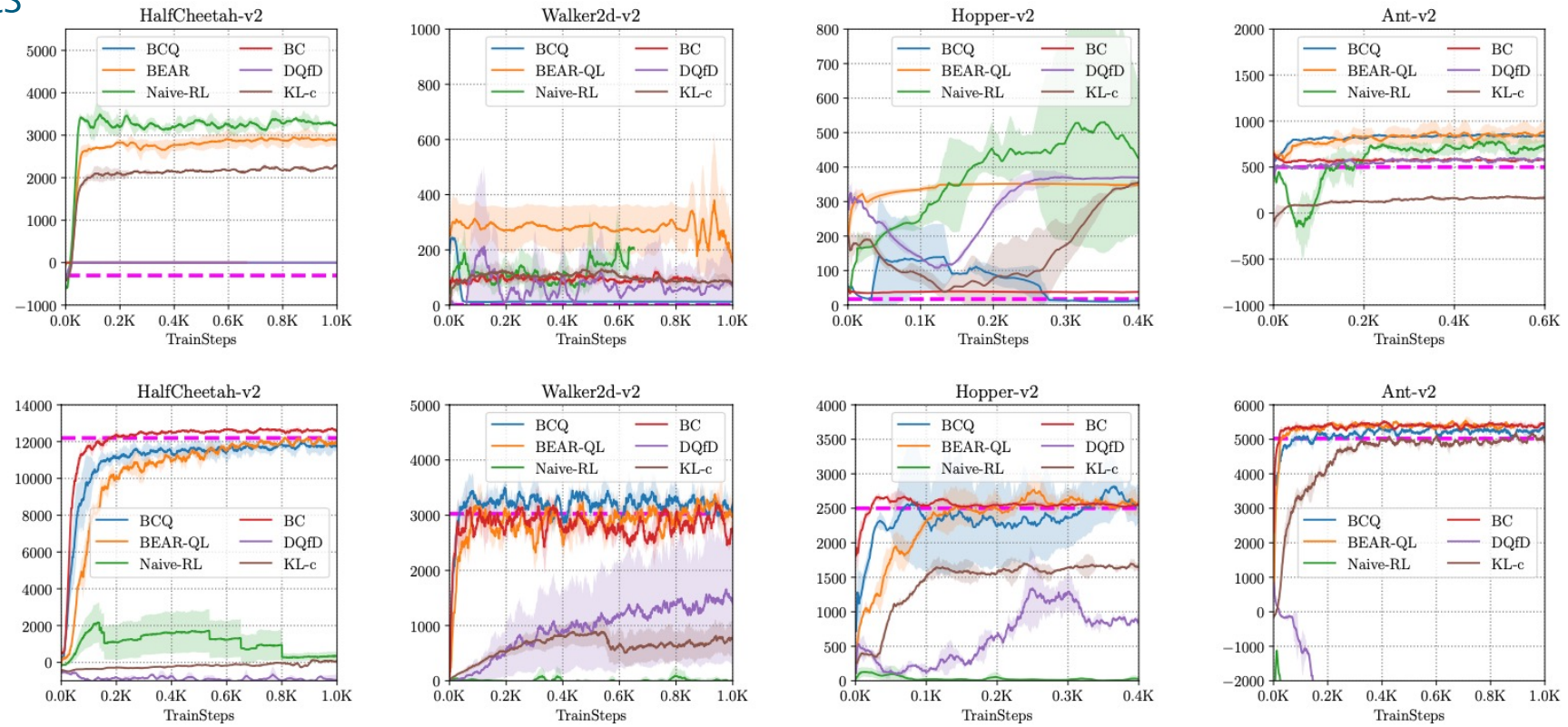


Figure 5: Average performance of BEAR-QL, BCQ, Naïve RL and BC on random data (top row) and optimal data (bottom row) over 5 seeds. BEAR-QL is the only algorithm capable of learning in both scenarios. Naïve RL cannot handle optimal data, since it does not illustrate mistakes, and BCQ favors a behavioral cloning strategy (performs quite close to behaviour cloning in most cases), causing it to fail on random data. Average return over the training dataset is indicated by the dashed magenta line.

Offline Reinforcement Learning

Conservative Methods

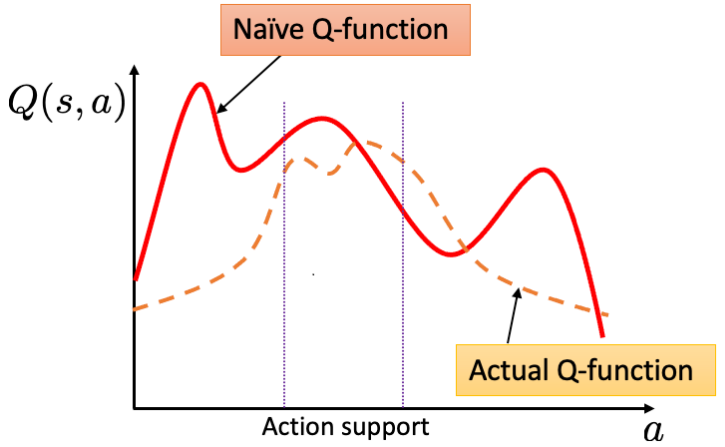
- What is Offline Reinforcement Learning?
- Challenges of Offline Reinforcement Learning
- Solution Approaches:
  - Policy-constrained Methods:
    - Batch-constrained Q-Learning (BCQ)
    - Bootstrapping Error Accumulation Reduction (BEAR)
  - **Conservative Methods:**
    - Conservative Q-Learning (CQL)
    - Model-based Offline Policy Optimization (MOPO)
    - Conservative Offline Model-Based Policy Optimization (COMBO)
- Summary

Conservative Offline RL

BCQ and BEAR – Uncertainty Perspective

- Different areas of the Q-Function are associated with different degrees of (un-)certainty
- BEAR and BCQ address the uncertainty by restricting the policy to certain areas where we think we are certain

How else could we address uncertainty?



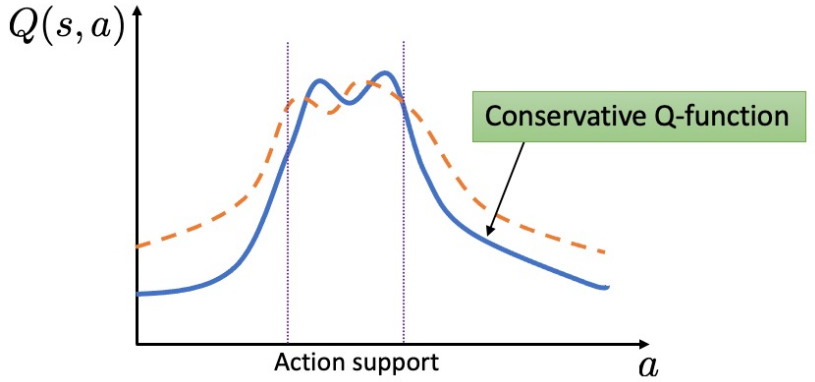
https://bair.berkeley.edu/blog/2020/12/07/offline/

Conservative Offline RL

Conservative Q-Learning

Idea of CQL[1]:

- Instead of constraining the action set for bootstrapping,…
- …be conservative in the estimation of state-action-values that are not in the dataset!
  - Tackles the problem in the policy evaluation by staying conservative in uncertain areas
  - *Implicitly* keeps the policy away from out of distribution actions



→ ***Learn a value function such that the estimated performance of the policy under this learned value function lower-bounds its true value!***

https://bair.berkeley.edu/blog/2020/12/07/offline/

[1] *Aviral Kumar et al.: Conservative Q-Learning for Offline Reinforcement Learning. NeurIPS. 2020.*

Conservative Offline RL

Conservative Evaluation

- **Goal:** find lower bound to Q_π using data $\mathcal{D}$ collected from π_β
- Let $\hat{T}_\pi$ be the empirical Bellman operator for policy π estimated from $\mathcal{D}$
- **CQL regularization**
  - Minimize $Q(s,a)$ on $s \in \mathcal{D}$, $a \sim \pi(s)$
  - For $a \sim \pi_\beta(s)$ there is no need to be conservative → additionally maximize
- CQL optimization problem for policy evaluation

$$\min_Q \alpha \cdot \left(\mathbb{E}_{s\sim\mathcal{D},a\sim\pi(s)}[Q(s,a)] - \mathbb{E}_{s,a\sim\mathcal{D}}[Q(s,a)]\right) + \frac{1}{2}\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[\left(Q(s,a) - \hat{T}_\pi Q(s,a)\right)^2\right],$$

where α is a trade-off parameter to control how conservative we are.

Conservative Offline RL

Conservative Control

- **Goal:** Find a good policy π from offline data $\mathcal{D}$ collected from π_β
- Iterate between policy improvement and conservative policy evaluation
- Online Conservative Q-Learning (π is the current policy)

$$\max_{\mu} \min_{Q} \left(\alpha \mathbb{E}_{s\sim\mathcal{D},a\sim\mu(a|s)}[Q(s,a)] - \alpha \mathbb{E}_{s,a\sim\mathcal{D}}[Q(s,a)]\right) + \frac{1}{2}\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[\left(Q(s,a) - \hat{T}^{\pi}\hat{Q}^{k}(s,a)\right)^2\right]$$

- Policy improvement can be done using an actor or standard Q-Learning

Conservative Offline RL

CQL - Performance

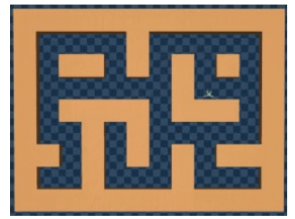
- Ant Maze:
  - Navigate from start to goal
  - Dataset consists of random motions and no single trajectory solves the task
  - The Offline RL algorithm needs to "stich" different sub-trajectories

D4RL Ant Maze	BC	SAC	BEAR	BRAC	AWR	BCQ	AlgaeDICE	CQL
U-Maze	65.0	0.0	73.0	70.0	56.0	**78.9**	0.0	74.0
U-maze + diverse	55.0	0.0	61.0	70.0	70.3	55.0	0.0	**84.0**
Medium + play	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**61.2**
Medium + diverse	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**53.7**
Large + play	0.0	0.0	0.0	0.0	0.0	6.7	0.0	**15.8**
Large + diverse	0.0	0.0	0.0	0.0	0.0	2.2	0.0	**14.9**

U-Maze

Medium

Large Maze



https://bair.berkeley.edu/blog/2020/12/07/offline/

Conservative Offline RL

Conservative Q-Learning

- It is proven that the return-estimate of the learned policy π under Q^{π}_{CQL} is a lower-bound on the actual policy performance:

$$J_{\mathrm{CQL}}(\pi) = \mathbb{E}_{s_0 \sim \mathrm{init}, a_0 \sim \pi}\left[Q^{\pi}_{\mathrm{CQL}}(s_0, a_0)\right] \leq E_{s_0 \sim \mathrm{init}, a_0 \sim \pi}[Q^{\pi}(s_0, a_0)] = J(\pi)$$

- We only need to add a regularizer term (which can be estimated from the dataset) during training
  - No need to estimate the behavior policy (as previous work needs)

Standard actor-critic algorithm

1. Learn $\hat{Q}^{\pi}$ using offline dataset $\mathcal{D}$.
2. Optimize policy: $\pi \leftarrow \arg\max_{\pi} \mathbb{E}_{\pi}[\hat{Q}^{\pi}]$.

CQL algorithm

1. Learn $\hat{Q}^{\pi}_{\mathrm{CQL}}$ using offline dataset $\mathcal{D}$.
2. Optimize policy: $\pi \leftarrow \arg\max_{\pi} \mathbb{E}_{\pi}[\hat{Q}^{\pi}_{\mathrm{CQL}}]$.

https://bair.berkeley.edu/blog/2020/12/07/offline/

Conservative Offline RL

Model-based Offline Policy Optimization

Model-based Offline Reinforcement Learning

1. Estimate $\mathrm{r}(s, a)$ and $p(s'|a, s)$ from $\mathcal{D}$
2. Apply active reinforcement learning to the model

- The transitions (s, a, r, s') can now be collected online → no distribution shift
- Nonetheless if π is very different from π_β our model is queried in unknown regions

→ We still need to address the uncertainty!

Conservative Offline RL

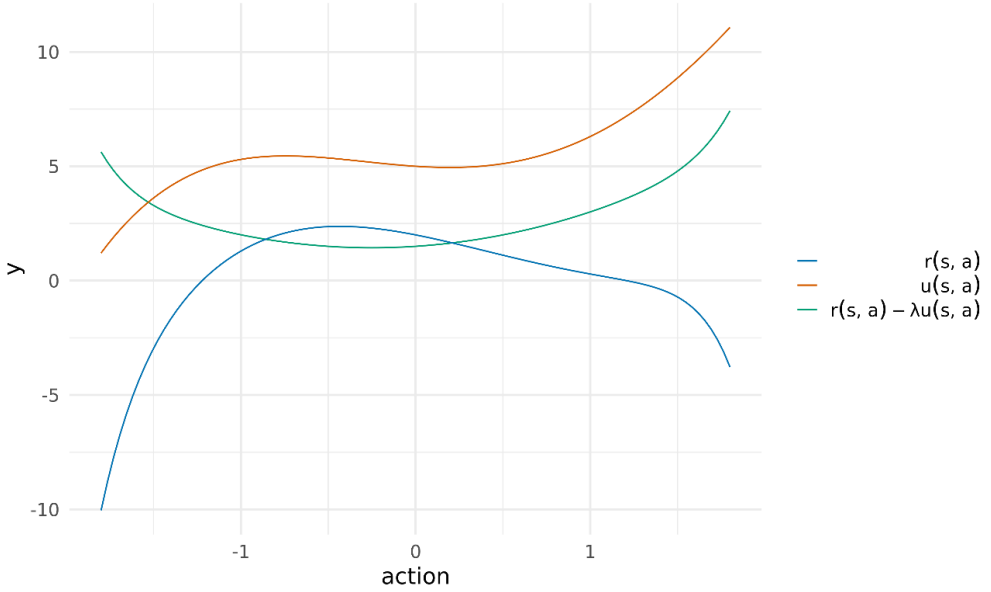
Model-based Offline Policy Optimization

- By Yu, Thomas, Yu, Ermon, Zou, Levine, Finn and Ma, 2020

Algorithm

1. Estimate $\mathrm{r}(s, a)$ and $p(s'|a, s)$ from $\mathcal{D}$
2. Learn uncertainty quantification $u(s, a)$ ← This is the hard part
3. Apply active reinforcement learning to the model using a pessimistic reward function

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

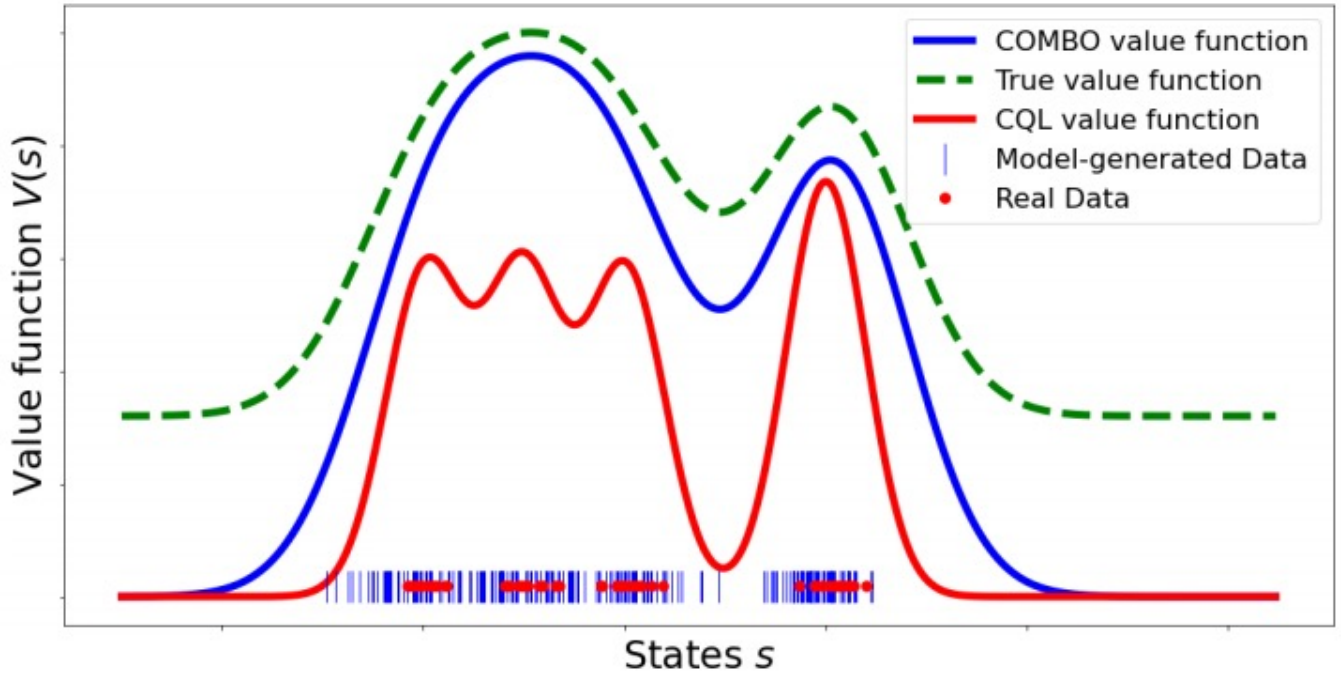


Fraunhofer IIS

Conservative Offline RL

Conservative Offline Model Based Policy Optimization

- by Yu, Kumar, Rafailov, Rajeswaran, Levine and Finn, 2021
- **Problem in MOPO:** Uncertainty quantification $u(s, a)$ is notoriously difficult for neural networks
- COMBO addresses this by combining CQL with rollouts from a model



Conservative Offline RL

Conservative Offline Model Based Policy Optimization

only for reference

CQL objective

$$\min_Q \left(\alpha \mathbb{E}_{s\sim\mathcal{D},a\sim\pi(s)}[Q(s,a)] - \alpha \mathbb{E}_{s,a\sim\mathcal{D}}[Q(s,a)]\right) + \frac{1}{2}\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[\left(Q(s,a) - \widehat{T}_\pi Q(s,a)\right)^2\right]$$

- Situation: Provided with buffer $\mathcal{D}$ and a model estimated from $\mathcal{D}$
- Let $\hat{d}_\pi$ be the marginal state distribution under π under our model
- Let $d_\pi^f(s,a)$ be the f-interpolation between the offline data and rollouts from the learned model
  - Sample from $\mathcal{D}$ with probability f
  - Sample from $d_\pi^f(s,a)$ with probability $1-f$

COMBO objective

$$\min_Q \left(\alpha \mathbb{E}_{s\sim d_\pi^f,a\sim\pi(s)}[Q(s,a)] - \alpha \mathbb{E}_{s,a\sim\mathcal{D}}[Q(s,a)]\right) + \frac{1}{2}\mathbb{E}_{s,a,s'\sim d_\pi^f}\left[\left(Q(s,a) - \widehat{T}_\pi Q(s,a)\right)^2\right]$$

- Note that we only maximize on $s,a\sim\mathcal{D}$ → Higher trust in original data than model rollouts!

Conservative Offline RL

Conservative Offline Model Based Policy Optimization

CQL objective

$$\min_Q \left(\alpha \mathbb{E}_{s\sim\mathcal{D},a\sim\pi(s)}[Q(s,a)] - \alpha \mathbb{E}_{s,a\sim\mathcal{D}}[Q(s,a)]\right) + \frac{1}{2}\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[\left(Q(s,a) - \hat{T}_\pi Q(s,a)\right)^2\right]$$

COMBO objective

$$\min_Q \left(\alpha \mathbb{E}_{s\sim d_\pi^f,a\sim\pi(s)}[Q(s,a)] - \alpha \mathbb{E}_{s,a\sim\mathcal{D}}[Q(s,a)]\right) + \frac{1}{2}\mathbb{E}_{s,a,s'\sim d_\pi^f}\left[\left(Q(s,a) - \hat{T}_\pi Q(s,a)\right)^2\right]$$

COMBO advantages over CQL

- Uses more data (i.e. from the model)
- The data is correlated with the policy

Conservative Offline RL

COMBO – Algorithm

only for reference

Algorithm 1 COMBO: Conservative Model Based Offline Policy Optimization

Require: Offline dataset $\mathcal{D}$, rollout distribution $\mu(\cdot|\mathbf{s})$, learned dynamics model $\widehat{T}_\theta$, initialized policy and critic π_ϕ and Q_ψ.

1: Train the probabilistic dynamics model $\widehat{T}_\theta(\mathbf{s}', r|\mathbf{s}, \mathbf{a}) = \mathcal{N}(\mu_\theta(\mathbf{s}, \mathbf{a}), \Sigma_\theta(\mathbf{s}, \mathbf{a}))$ on $\mathcal{D}$.
2: Initialize the replay buffer $\mathcal{D}_{\text{model}} \leftarrow \varnothing$.
3: **for** $i = 1, 2, 3, \cdots$, **do**
4: Perform model rollouts by drawing samples from μ and $\widehat{T}_\theta$ starting from states in $\mathcal{D}$. Add model rollouts to $\mathcal{D}_{\text{model}}$.
5: Conservatively evaluate current policy by repeatedly solving eq. 4 to obtain $\hat{Q}_\psi^{\pi_\phi^i}$ using data sampled from $\mathcal{D} \cup \mathcal{D}_{\text{model}}$.
6: Improve policy under state marginal of d_f by solving eq. 5 to obtain π_ϕ^{i+1}.
7: **end for**

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

$$\pi' \leftarrow \arg\max_\pi \mathbb{E}_{\mathbf{s}\sim\rho,\mathbf{a}\sim\pi(\cdot|\mathbf{s})}\left[\hat{Q}^\pi(\mathbf{s},\mathbf{a})\right] \quad (5)$$

Conservative Offline RL

COMBO – Experimental Results

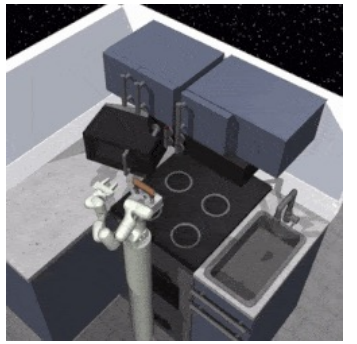
only for reference

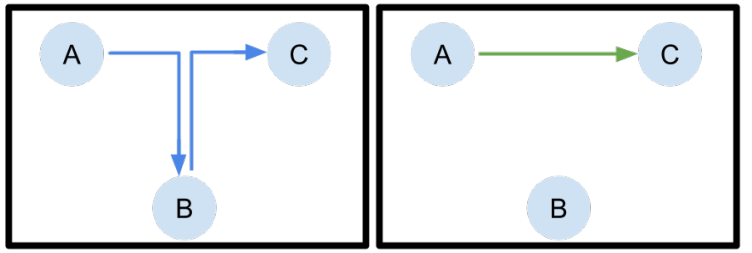
Dataset type	Environment	BC	COMBO (ours)	MOPO	CQL	SAC-off	BEAR	BRAC-p	BRAC-v
random	halfcheetah	2.1	**38.8**	35.4	35.4	30.5	25.1	24.1	31.2
random	hopper	1.6	**17.9**	11.7	10.8	11.3	11.4	11.0	12.2
random	walker2d	9.8	7.0	**13.6**	7.0	4.1	7.3	-0.2	1.9
medium	halfcheetah	36.1	**54.2**	42.3	44.4	-4.3	41.7	43.8	46.3
medium	hopper	29.0	**94.9**	28.0	86.6	0.8	52.1	32.7	31.1
medium	walker2d	6.6	75.5	17.8	74.5	0.9	59.1	77.5	**81.1**
medium-replay	halfcheetah	38.4	**55.1**	53.1	46.2	-2.4	38.6	45.4	47.7
medium-replay	hopper	11.8	**73.1**	67.5	48.6	3.5	33.7	0.6	0.6
medium-replay	walker2d	11.3	**56.0**	39.0	32.6	1.9	19.2	-0.3	0.9
med-expert	halfcheetah	35.8	**90.0**	63.3	62.4	1.8	53.4	44.2	41.9
med-expert	hopper	111.9	**111.1**	23.7	111.0	1.6	96.3	1.9	0.8
med-expert	walker2d	6.4	96.1	44.6	**98.7**	-0.1	40.1	76.9	81.6

Offline RL / Summary

Why Offline RL Should Work in Principle

- Find the good behaviour in a dataset of good and bad behaviour
- Generalize to similar states and actions
- Stich together good and bad behaviour

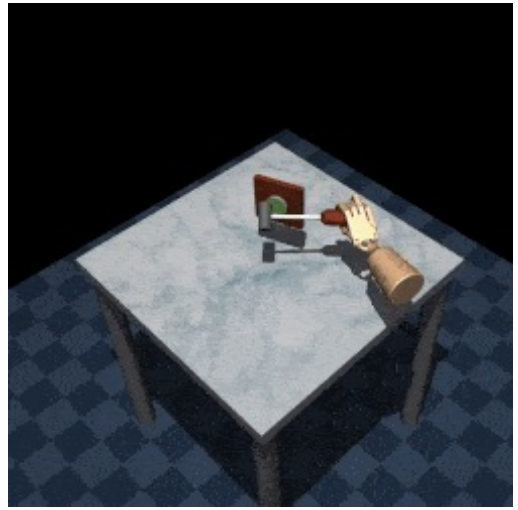




https://bair.berkeley.edu/blog/2020/06/25/D4RL/

AntMaze

Carla



Adroit

Offline Reinforcement Learning

References

Books

- Sutton, Richard and Barto, Andrew: Reinforcement Learning. 2015

Scientific Articles and Tutorials

- Fujimoto, Scott and Meger, David and Precup, Doina.: Off-Policy Deep Reinforcement Learning without Exploration. 2019
- Kumar, Aviral and Fu, Justin and Tucker, George and Levine, Sergey.: Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. 2019.
- Kumar, Aviral and Zhou Aurick and Tucker, George and Levine, Sergey: Conservative Q-Learning for Offline Reinforcement Learning. 2020
- Yu, Tianhe and Thomas, Garrett and Yu, Lantao and Ermon, Stefano and Zou, James and Levine, Sergey and Finn, Chelsea and Ma Tengyu. MOPO: Model-based Offline Policy Optimization. 2020
- Liu, Yao and Swaminathan, Adith and Agarwal, Alekh and Brunskill, Emma: Off-Policy Policy Gradient with State Distribution Correction. 2019
- Yu, Tianhe and Kumar, Aviral, and Rafailov, Rafael and Rajeswaran, Aravind and Levine, Sergey and Finn, Chelsea: COMBO: Conservative Offline Model-Based Policy Optimization. 2021
- Agarwal, Rishabh; Schuurmans, Dale; Norouzi, Mohammad: An Optimistic Perspective on Offline Reinforcement Learning. 2020.
- Levine, Sergey and Kumar, Aviral and Tucker, George and Fu, Justin: Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. 2020.
- Fujimoto, Scottt and Conti, Edoardo and Ghavamzadeh, Mohammad and Pineau, Joelle: Benchmarking Batch Deep Reinforcement Learning Algorithms. 2019.
- Fujimoto, Scott and van Hoof, Herke and Meger, David: Addressing Function Approximation in Actor-Critic Methods. 2018.

Websites

- Kumar, Aviral and Singh, Avi: Offline Reinforcement Learning: How Conservative Algorithms can Enable New Applications. Accessed May 2021.
- Kumar, Aviral: Data-Driven Deep Reinforcement Learning. Accessed May 2021.
- Levine, Sergey: Decisions from Data: Hoe Offline Reinforcement Learning Will Change How We Use Machine Learning. Accessed May 2021.
- Hui, Jonathan: RL – Dual Gradient Descent. Accessed May 2021.
- Stack Exchange: Maximum Mean Discrepancy. Accessed May. 2021.