Offline Reinforcement Learning

Slides developed from Prudencio et al. "A Survey on Offline Reinforcement Learning: Taxonomy, Review and Open Problems"

Bryce MacInnis

Offline Reinforcement Learning

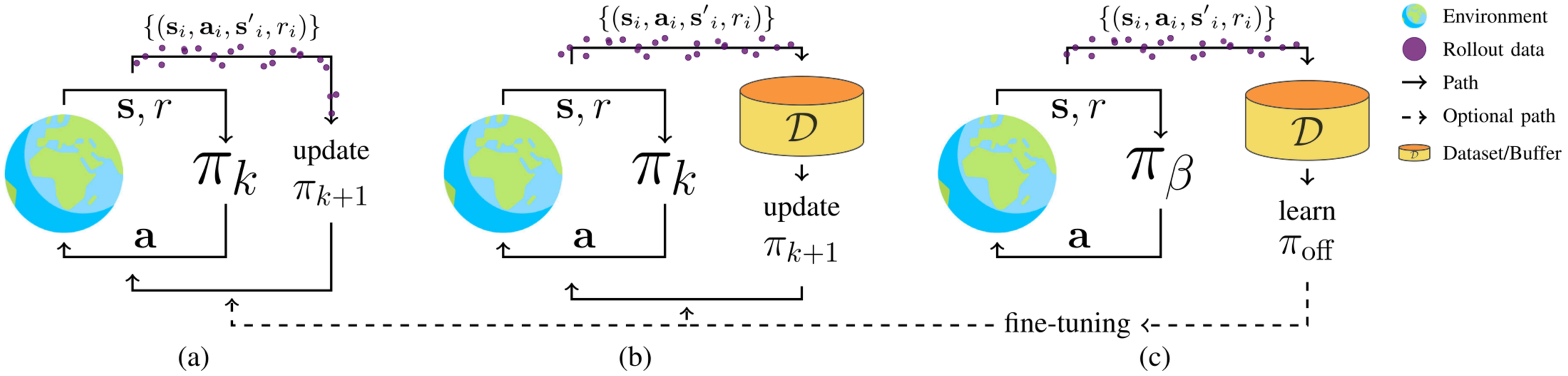


Figure taken from Prudencio et al. "A Survey on Offline Reinforcement Learning: Taxonomy, Review and Open Problems"

In Online Reinforcement Learning, every time you make a change to the policy you need to collect new data.

Offline Reinforcement Learning

Applications

- **Healthcare settings**: Offline RL can optimize treatment, medicine rates using historical patient data without experimenting on patients.
- **Autonomous vehicles:** Using pre-collected data directly from the sensors of the vehicle rather than learning from initially random behaviour (or fresh out of the simulator)
- **Robotics:** Controllers are easily damaged from locomotion failures, pre-training from a dataset collected from simulators may prevent damage.

Offline Reinforcement Learning

Motivation

- Reinforcement Learning isn't widely adopted in industry yet because of: safety concerns, cost of collecting data, challenge of modelling dynamic systems as simulators.
- Safety-critical RL requires pre-training to mind the gap from the simulator to the real-world.
- Collecting data can be expensive or infeasible depending on the application, online RL is inherently wasteful (low sample efficiency).
- Building simulators that can model the complexities of real-life is difficult (alignment gap).

In off-policy RL, we've already gone through the effort of collecting a dataset/replay buffer. Why can't we learn from that directly?

Offline Reinforcement Learning

$$\mathscr{D} = (s, a, r, s') \sim \beta(a \mid s)$$

$$\pi(a \mid s) = \arg\max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{T} \gamma^t r_t \right] \quad \text{s.t. } \pi \text{ is learned from } \mathscr{D}$$

Why not off-policy RL for offline RL?

- A replay buffer and an offline dataset are roughly equivalent, so the off-policy RL methods should work for offline RL right?
- Off-policy RL methods are likely to produce OOD actions that are positively biased.
- In Offline RL, you can't get feedback from the environment so these OOD actions are included in the policy and never evaluated by the environment.

Offline Reinforcement Learning

Challenges

- Can't correct bad behaviours without interacting with the environment
- Dealing with stochasticity and non-stationary environments
- Over-fitting vs under-fitting (bias-variance trade-off)
- Distributional shift

But first, a detour to look at Actor-Critic networks in more detail

Policy Gradients

Motivation

- DQN learns a Q-value for each state, action pair. Generating a policy is easy: choose whatever action maximizes the cumulative reward.
- What about for continuous actions? Which action (of an infinite possibilities) should we choose?
- Do we sample the action space and choose the best?
- Policy gradients directly optimize the policy (i.e. which actions to choose).

DDPG - Deep Deterministic Policy Gradient

Lillicrap et al. "Continuous control with deep reinforcement learning" (2015)

- Actor-Critic Architecture
  - Actor learns a deterministic policy π(s) that maximizes the Q-value.

$$\pi(s) = \arg\max_a Q(s, a)$$

  - Critic tries to minimize the TD error:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')}\left[\left(Q(s, a) - \left(r + \gamma Q(s', \pi(s'))\right)\right)^2\right]$$

DDPG - Deep Deterministic Policy Gradient

Problems

- The Q-values are a neural network (function approximator) which means that the Q-values are imperfect (positive bias)
- Because the actor is trained to maximize Q, the actor is going to exploit the imperfection in the Q-values / Critic network.
- These errors compound over time leading to poor performance.

TD3 - Twin Delayed DDPG

Fujimoto et al. "Addressing Function Approximation Error in Actor-Critic Methods" (2018)

1. Two critic networks: the conservative estimate of the Q-value is chosen.

$$\mathscr{L}_{\text{critic}} = \mathbb{E}_{(s,a,r,s')\sim\mathscr{D}}\left[\left(Q_1(s,a)-y\right)^2+\left(Q_2(s,a)-y\right)^2\right]$$

$$y = r + \gamma \cdot \min_{i=1,2} Q_i(s', \pi(s') + \epsilon)$$

$$\epsilon \sim \text{clip}(\mathcal{N}(0,\sigma), -c, c)$$

2. Clipped noise is added to the target action for smoothing

3. Actor network is updated less than the critic network

What is Advantage?

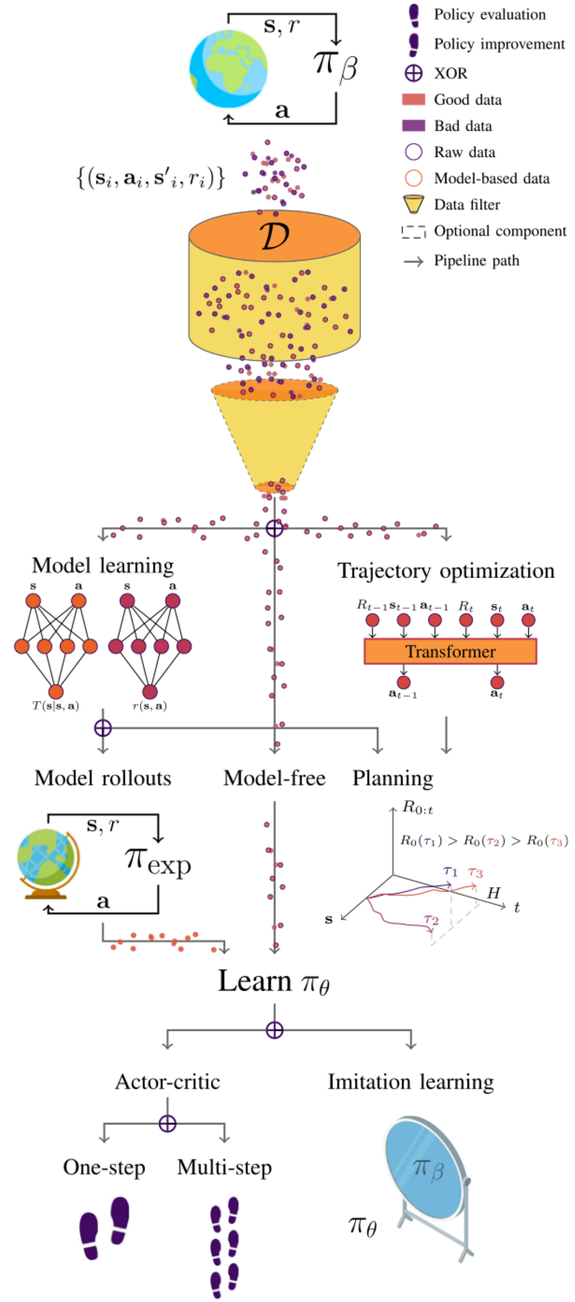
- The **advantage function** measures how much better (or worse) an action is compared to the average action at that state.

$$A(s, a) = Q(s, a) - V(s)$$

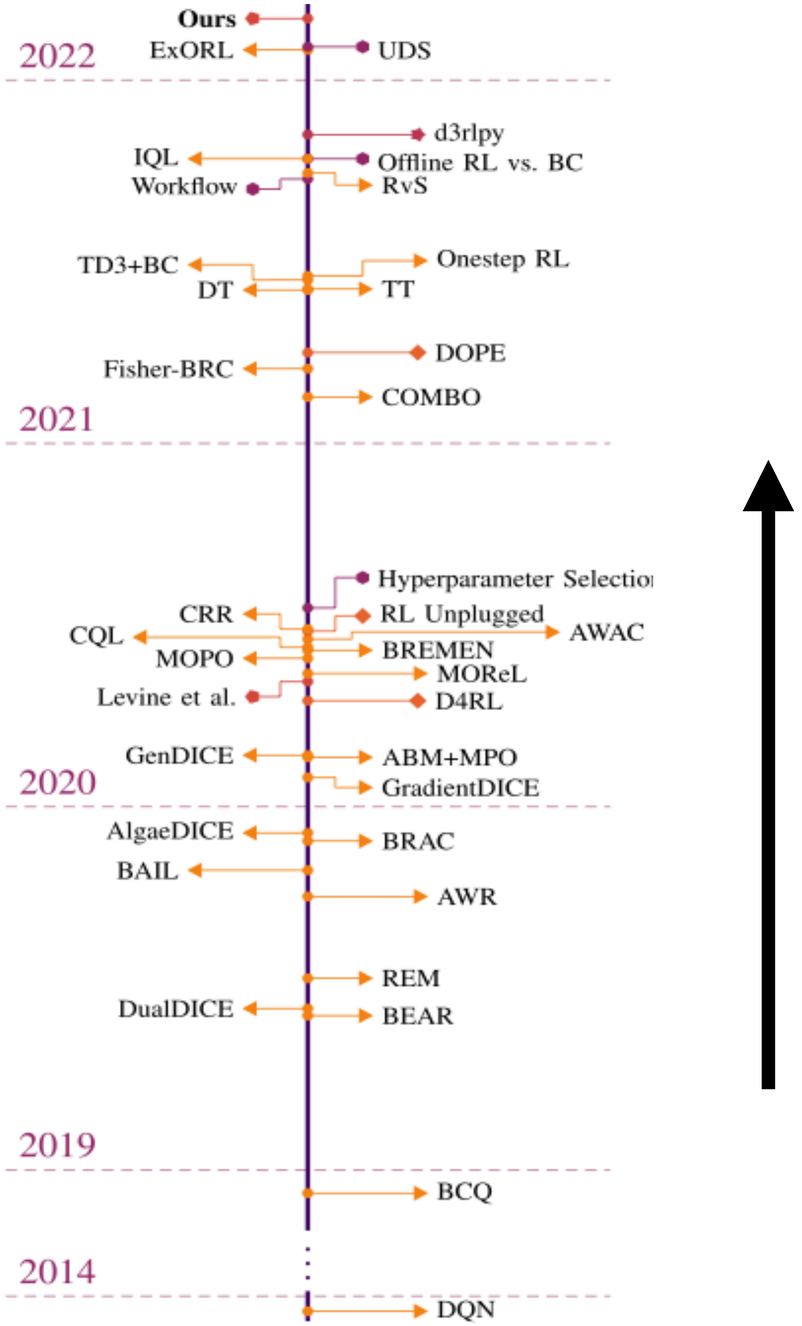
- Why use it?
  - Variance reduction: Helps stabilize learning by centering Q-values
  - Emphasizes actions that are better than average.

Offline Reinforcement Learning

- Policy constraints
- Policy regularization
- Model-based Offline RL
- Uncertainty Estimation
- Importance Sampling
- One-step methods
- Imitation Learning
- Trajectory Optimization



Development Timeline



Policy Constraints

- **Goal: Keep the learned policy close to the dataset to avoid out-of-distribution actions.**
- Variants:
  - Direct Constraints: Add a penalty for divergence from the behaviour policy.

$$D_{\text{KL}}(\pi(a \mid s) \;||\; \beta(a \mid s))$$

  - Implicit Constraints: Use MSE, advantage weighting, or other losses to naturally bias the policy away from OOD actions.
- Can be too pessimistic: "if we know that a certain state has all actions with zero reward, we should not care about constraining the policy."

Fitted Q-Iteration

Damien Ernst, Pierre Geurts, Louis Wehenkel (2005) "Tree-Based Batch Mode Reinforcement Learning"

- Reformulate Offline RL as a regression problem.

$$Q_n(s,a) = r + \gamma \max_{a'} Q_{n-1}(s',a')$$

$$Q_0(s,a) = 0$$

- The policy is determined by:

$$\pi(s) = \arg\max_{a} Q(s,a)$$

- Discrete actions only

You'll hear about this one again

BCQ - Batch Constrained Q-Learning

Scott Fujimoto, David Meger, Doina Precup (2018) "Off-Policy Deep Reinforcement Learning without Exploration"

- Learn a generative model (VAE) of the dataset's actions
- Sample potential actions from the model
- Choose the action with the highest Q-value among them
- Avoids Out-of-Distribution actions

BCQ - Batch Constrained Q-Learning

The Issue

- The policy is over-conservative i.e. performance is only as good as the demonstrated policy in the offline dataset.
- If we know the demonstrated policy in the offline dataset has poor performance in certain regions of the search space, should we still constrain it?

BEAR - Bootstrapping Error Reduction

Aviral Kumar, Justin Fu, George Tucker, Sergey Levine (2019) "Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction "

- BEAR directly constrains the policy to stay within the support of the dataset.
- BCQ has flaws: it relies on the generative model that produces "plausible" actions but not truly in-distribution points. It samples the action space.
- BEAR fixes this by: learning a stochastic Gaussian policy, uses MMD metric to measure similarity between the learned policy and the policy in the dataset.
- Penalizes deviation from the demonstrated policy.

BRAC - Behaviour Regularized Actor-Critic

Yifan Wu, George Tucker, Ofir Nachum (2019)

- Keeping the same policy as the original dataset is useful for the actions with high Q-values but if the original policy isn't good in the search region, deviate from the policy.
- Train an Actor-Critic network, estimate the advantage of each action. Imitate the actions with high advantage.
- Two variants: BRAC-p, BRAC-v.
  - BRAC-p: Modifies the objective to penalize divergence
  - BRAC-v: Uses the advantage-weighted imitation loss

TD3+BC

Scott Fujimoto, Shixiang Shane Gu (2021) "A Minimalist Approach to Offline Reinforcement Learning"

- Offline Reinforcement Learning has gotten too complicated: what's the least amount of effort or engineering that needs to be put into it.
- Two changes to TD3, everything else is the same:
  - Given that the dataset is offline, we know the mean and std of the data and therefore we can Z-score normalize it (easier function approximation).
  - Add a regularization term (MSE) to penalize deviations from the actions taken by the policy in the dataset.
- Basically today's state-of-the-art method.
- Lambda controls the effect of the regularization, 0 lambda is vanilla TD3, 1 lambda is behavioural cloning.

Regularization

- Adding penalty terms that aren't directly related to constraining the policy to the behavioural policy.
- Typically used to add pessimism to the Critic network

CQL: Conservative Q-Learning

Aviral Kumar, Aurick Zhou, George Tucker, Sergey Levine (2020) "Conservative Q-Learning for Offline Reinforcement Learning

- **Goal: Make Q-values smaller for actions not seen in the dataset, so the policy avoids choosing them.**

$$\mathscr{L}_{\text{CQL}} = \mathbb{E}_{(s,a,r,s')\sim\mathscr{D}} \left[\left(Q(s,a) - \left(r + \gamma \max_{a'} Q(s',a') \right) \right)^2 \right]$$

$$+\alpha \cdot \left(\mathbb{E}_{a\sim\pi(\cdot|s)}[Q(s,a)] - \mathbb{E}_{a\sim\mathscr{D}}[Q(s,a)] \right)$$

- A penalty to push down Q-values of actions outside of the dataset.

Uncertainty Estimation

- Uncertainty-based methods allow switching between conservative and naive off-policy RL methods based on how much we trust the model.
- If we're in a low-uncertainty region (lots of support from the dataset), we can relax the policy constraints.
- If we're in a high-uncertainty region (limited data coverage), we constrain the policy.

REM: Random Ensemble Mixture

Rishabh Agarwal, Dale Schuurmans, Mohammad Norouzi (2020) "An Optimistic Perspective on Offline Reinforcement Learning"

- Sample a random convex combination of Q-functions and use them to estimate the Q-value.

$$Q^{\text{REM}}(s, a; \mathbf{w}) = \sum_{i=1}^{K} w_i Q_i(s, a)$$

$$\text{where} \quad w_i \geq 0, \quad \sum_{i=1}^{K} w_i = 1$$

$$TDError = r + \gamma \max_{a'} Q_{\text{REM}}(s', a')$$

One-Step Methods

- Motivation: Avoid bootstrapping errors (like TD's overestimation bias)
- Strategy:

  1. Train $Q(s, a)$ on a dataset using a fitted Q-iteration method
  2. Freeze the $Q(s, a)$ network
  3. Train actor **once** by maximizing the reward through conventional methods.

IQL: Implicit Q-Learning

Ilya Kostrikov, Ashvin Nair, Sergey Levine (2021) Offline Reinforcement Learning with Implicit Q-Learning

- Train $Q(s, a)$ using the loss function:

$$\mathscr{L}_Q = \Big(Q(s,a) - \big[r + \gamma V(s')\big]\Big)^2$$

- Train $V(S)$ using the loss function:

$$\mathscr{L}_V = \left(V(s) - \mathbb{E}_{a\sim\beta(\cdot|s)}\left[\exp\left(\frac{Q(s,a) - V(s)}{\beta}\right) Q(s,a)\right]\right)^2$$

- Compute the advantage $A(s, a) = Q(s, a) - V(s)$
- Train $\pi(a \,|\, s)$ using the loss function:

$$\mathscr{L}_\pi = -\log \pi(a \mid s) \cdot \exp\left(\frac{Q(s,a) - V(s)}{\alpha}\right)$$

Imitation Learning

- Does not use reward information, simply tries to make the same predictions given an action as the critic.
- Subject to distributional shift.
- Often successful if the dataset consists of expert behaviour
- Objective: Filter out suboptimal data and then imitate the expert data in the dataset.

RvS: Offline RL via Supervised Learning

Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, Sergey Levine (2021) "RvS: What is Essential for Offline RL via Supervised Learning?"

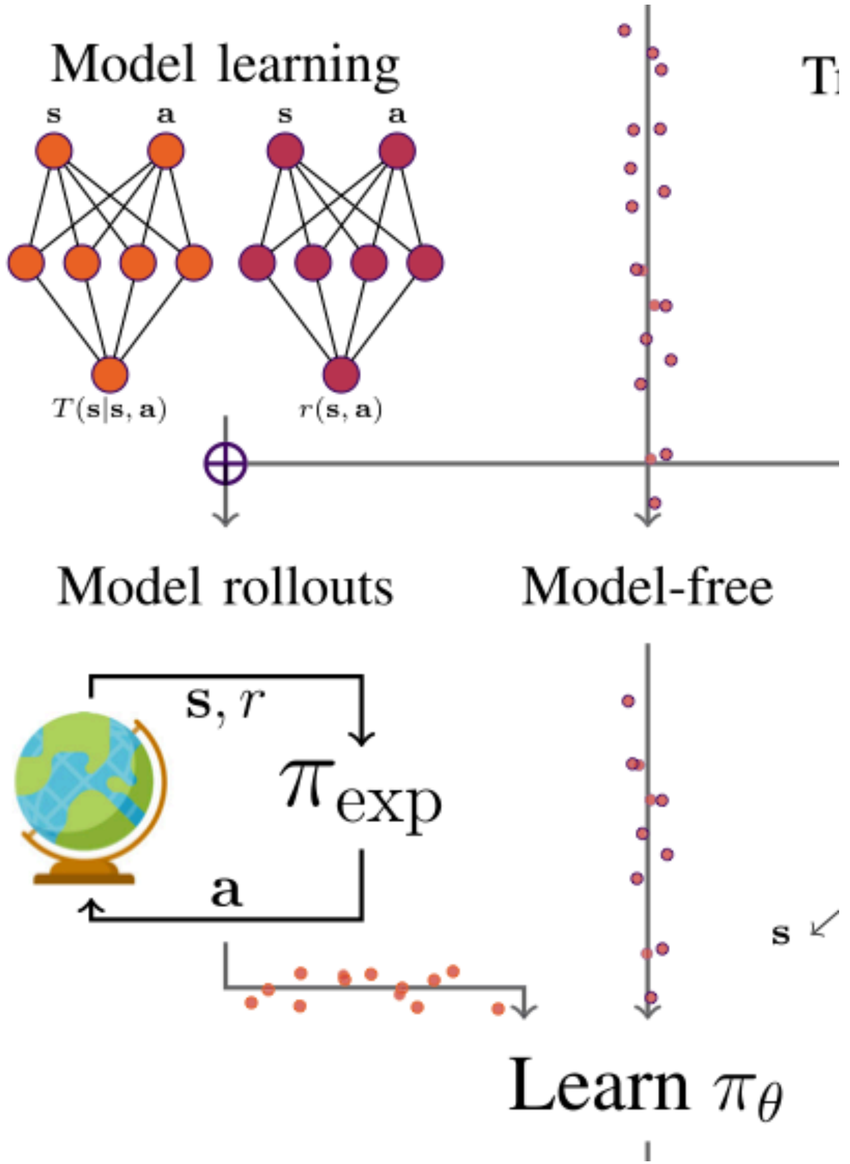
- The **Return-to-go** is an included input feature
- Otherwise, it's a behavioural cloning supervised learning regression problem.

$$\mathscr{L}(\theta) = \mathbb{E}_{(s_t, a_t, G_t) \sim \mathscr{D}} \left[\left\| \pi_\theta(s_t, R_t) - a_t \right\|^2 \right]$$

- Doesn't extrapolate or improve beyond the policy demonstrated in the dataset
- Works best with expert data: fails on suboptimal quality datasets.

Model-based Methods

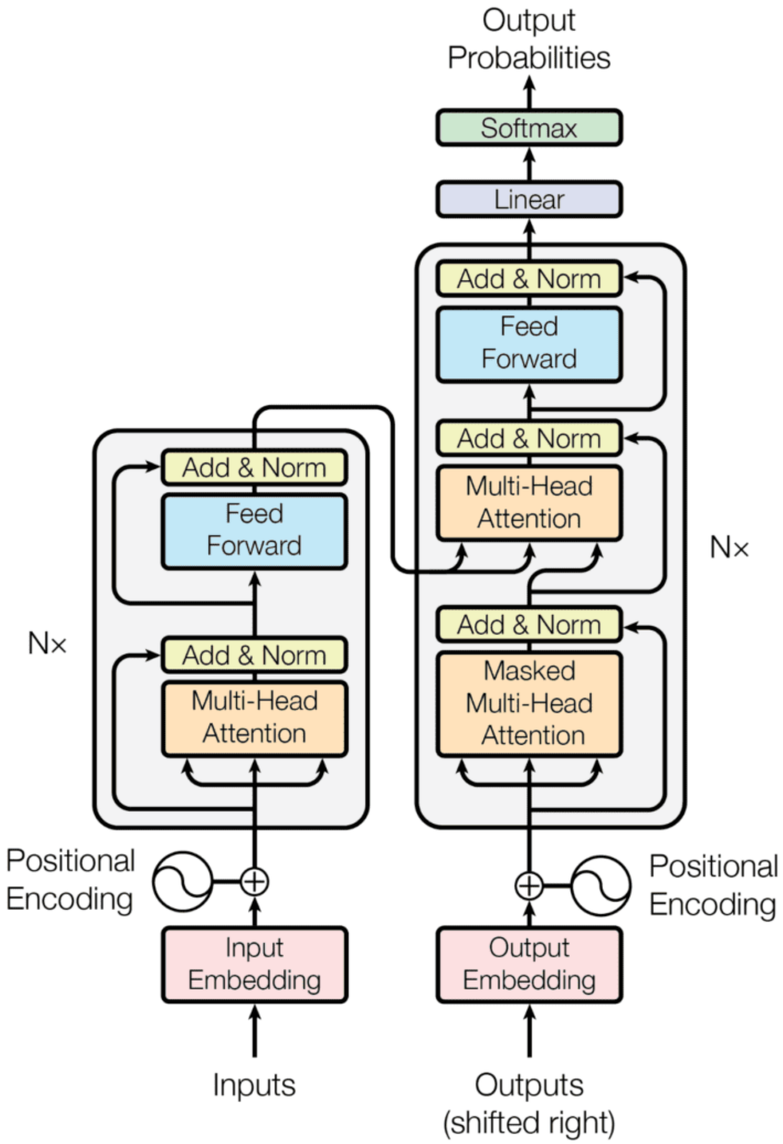
- Learn a model of the environment from the offline dataset, use the model to simulate online learning.
- Uses supervised learning to learn a dynamics model $T_{(s|s,a)}$ and $r(s,a)$
- Use policy gradients, imitation learning, etc to learn the policy.
- Depends on the quality and coverage of the model.



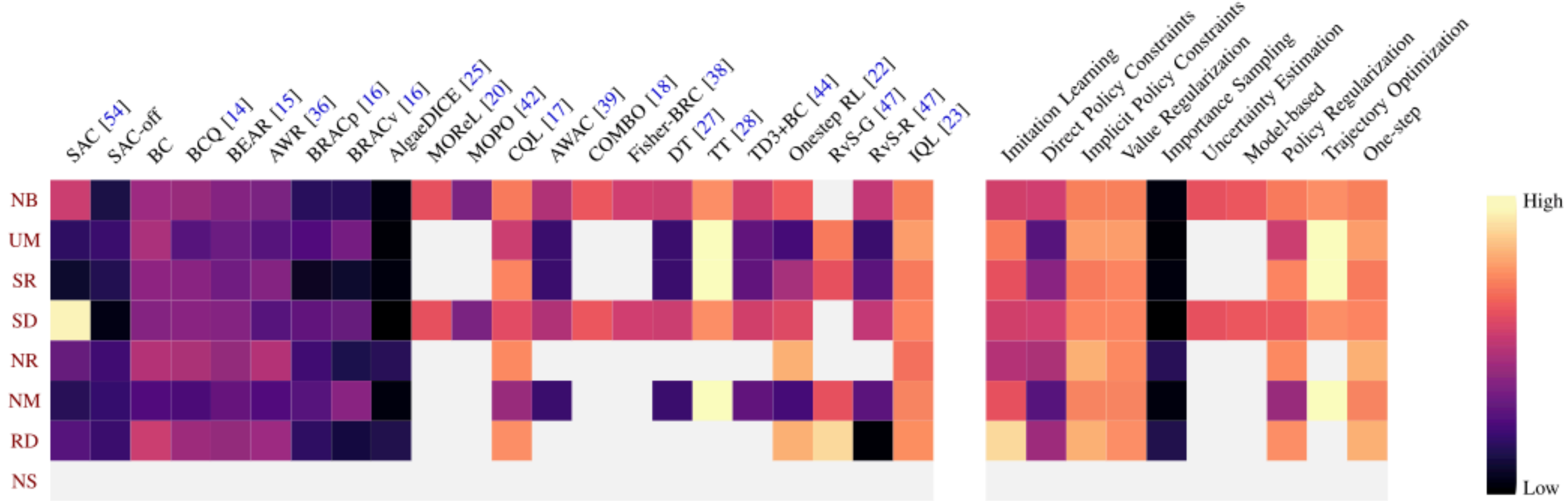
Trajectory Optimization Methods

Decision Transformer

- Reframe Reinforcement Learning as a Sequence Modelling task
- Treat trajectories like sentences and predict the next action like the next word.
- Based on Transformer architectures e.g. positional encoding, multi-head attention, etc.
- (RTG, s_0, [PAD]) -> a_0
- (RTG_1, s_1, a_0) -> a_1
- (RTG_2, s_2, a_1) -> a_2



Performance of Methods



Choosing an Offline RL method

You have **expert data** only?	**RvS**
You have **mixed-quality** data?	**IQL**, **CQL**
Want a **simple**, ready-to-go model?	**TD3+BC**

Off-policy Evaluation

- Evaluating the performance of Online RL is easy.
- Interact with the environment and then observe the reward.
- How do we evaluate the performance of an offline policy WITHOUT interacting with the environment?
- This is essential for hyper-parameter tuning and benchmarking offline RL methods.

Importance Sampling

Why do we need it?

- In Offline RL, our data comes from a behaviour policy
- But we want to learn a new policy Pi(A|S)
- Mismatch Problem: We don't have data from our learned policy, how can we estimate what the data would look like under Pi(A|S)

Importance Sampling

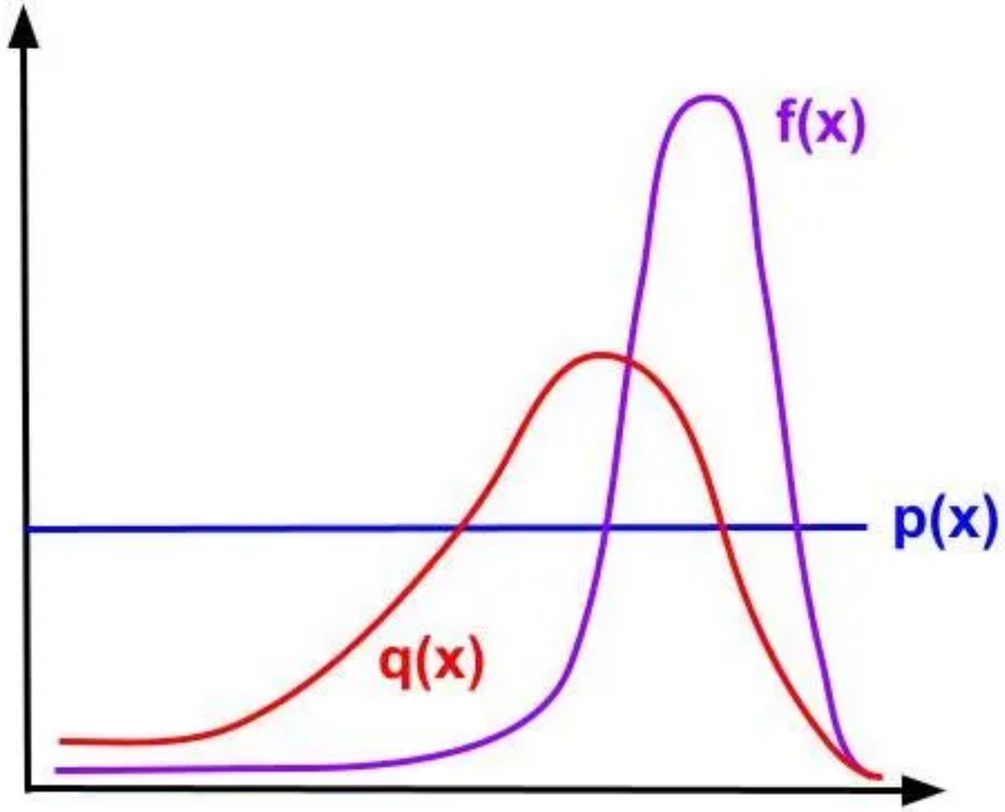
- Trick: Reweight samples collected under the behavioural policy to approximate expectations under Pi(S|A)

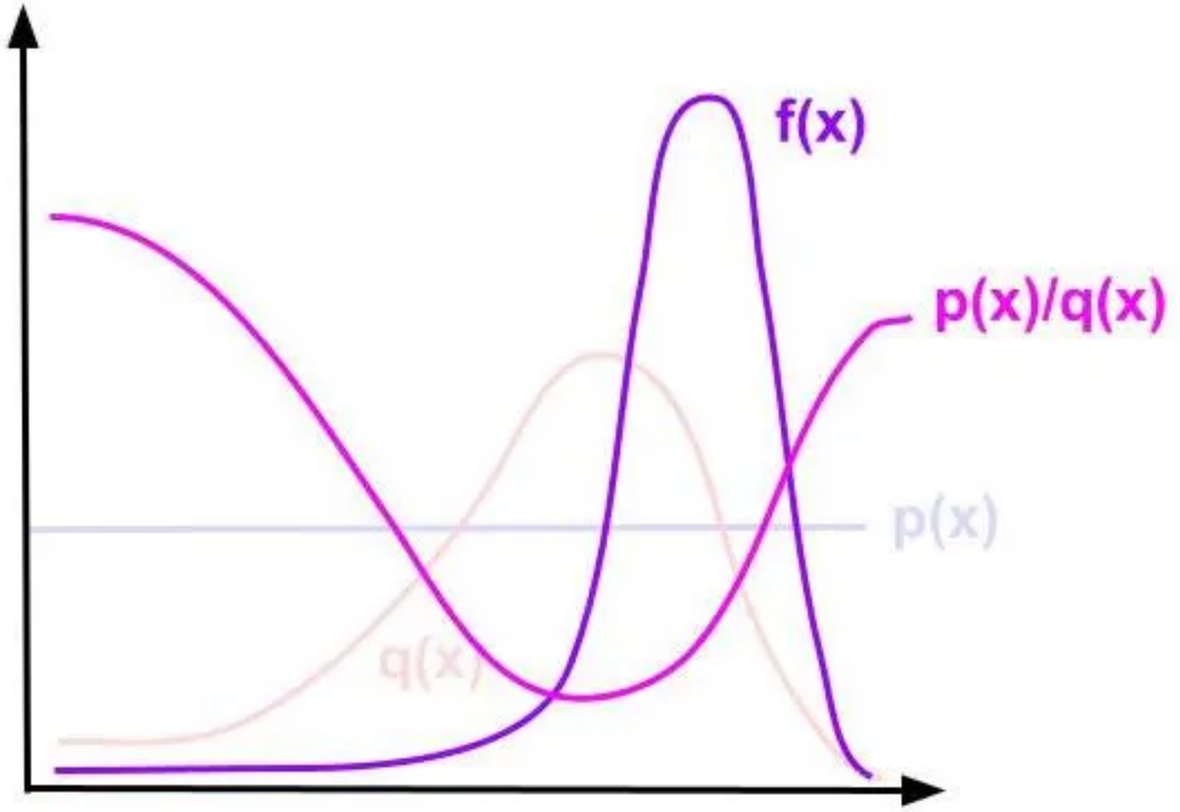
$$w(s,a) = \frac{\pi(a \mid s)}{\beta(a \mid s)}$$

$$\hat{V}^{\pi}(s) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[w(s,a) \cdot Q(s,a)\right]$$

Importance Sampling

A Geometric Interpretation





Off-policy Evaluation

- Three existing methods:
    1. Model-based Evaluation
    2. Importance Sampling
    3. Fitted Q-Evaluation

Fitted Q-Evaluation

- Offline Q-learning from the dataset i.e. train a Q-network (typically a neural network)
- Use final $Q^{\pi}(s, a)$ to estimate each state, action pair of the learned policy.
- The value of the policy:

$$V^{\pi}(s) = E_{a \sim \pi} Q^{\pi}(s, a)$$

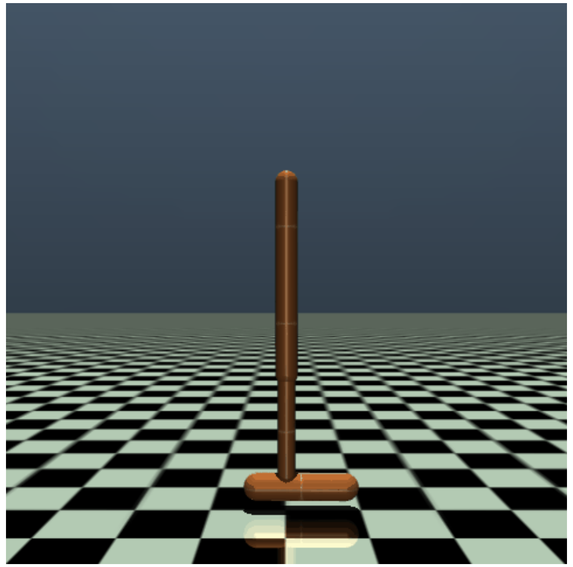
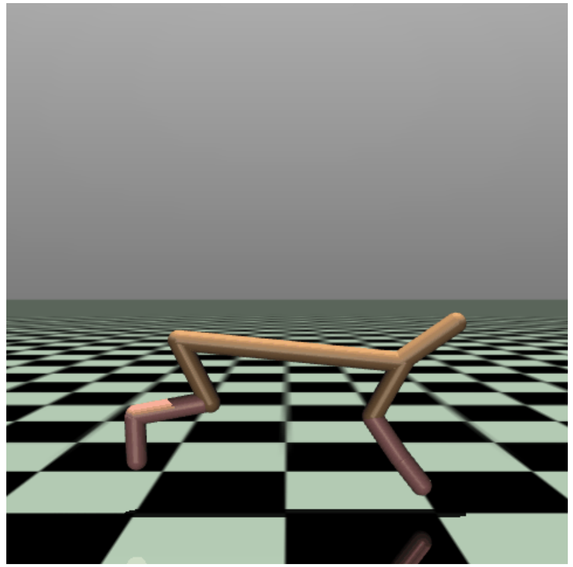
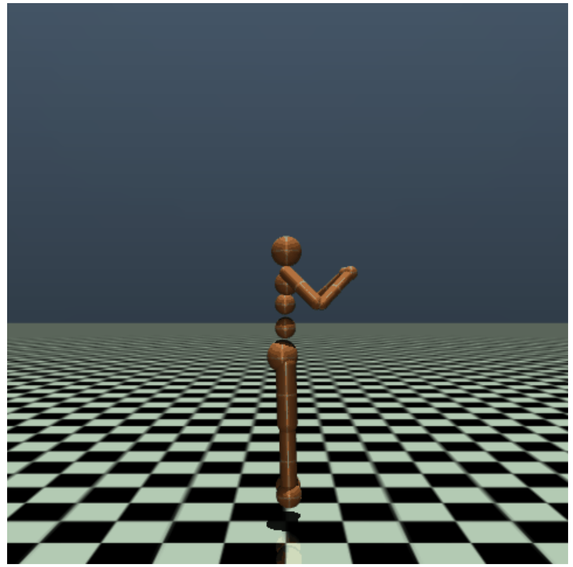
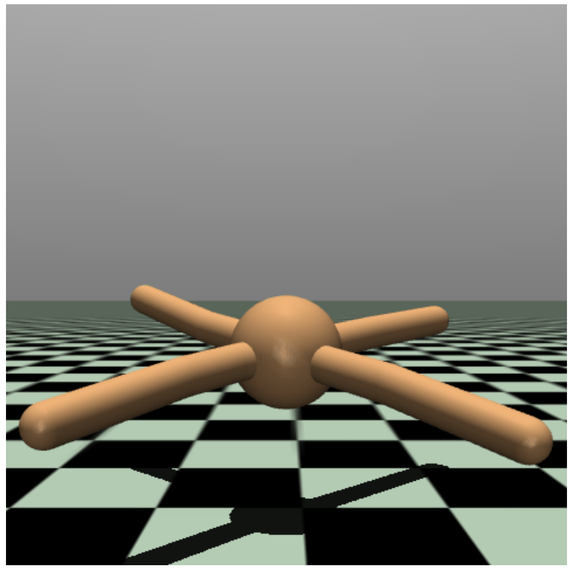
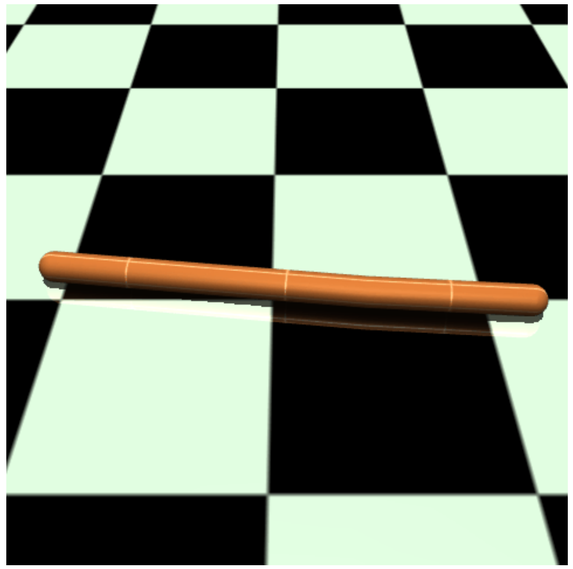
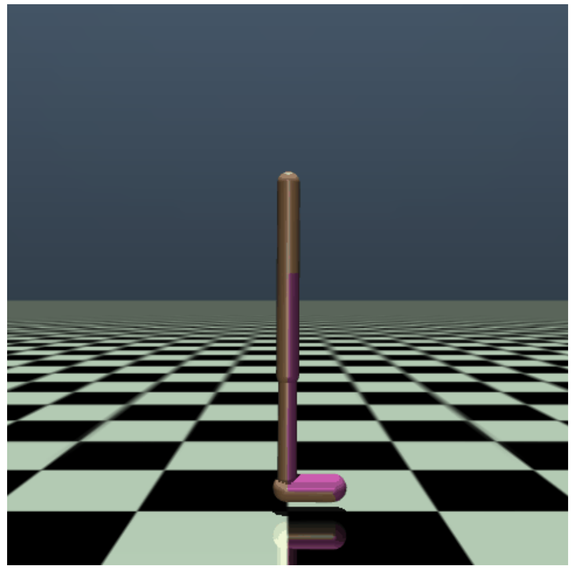
- Empirically, the best method for Offline Policy Evaluation right now.

Benchmarks

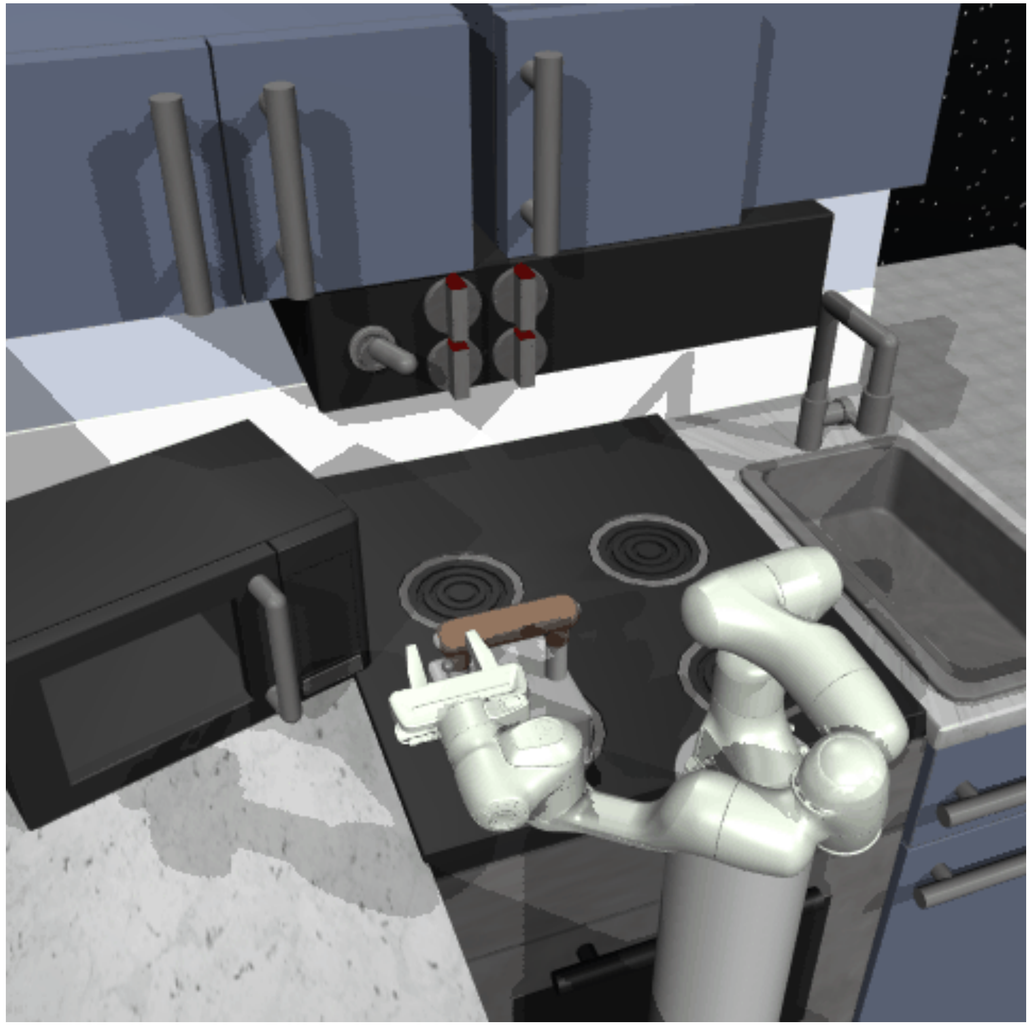
What makes a good Offline RL benchmark?

- Narrow and Biased Distributions: When data is limited or done only a certain way in human demonstrations, it's important that Offline RL is robust to this.
- Undirected Tasks: Assesses the ability of the method to stitch trajectories. I.e. solving a task even if none of the individual trajectories are complete solutions.
- Sparse Rewards: Difficult to solve due to the nature of the Bellman equation -- how do you bootstrap without a reward signal
- Suboptimal Data: How does the method perform when there are not expert demonstrations but instead low-quality trajectories.
- Realistic Domains: In order to assess real-world applicabilities, benchmarks should model complex systems as close to reality as possible.
- Nonstationarity: In Reinforcement learning, sometimes the distribution of the task changes and what works no longer does -- or occasionally there is an element of luck. Offline RL methods should be robust to this.

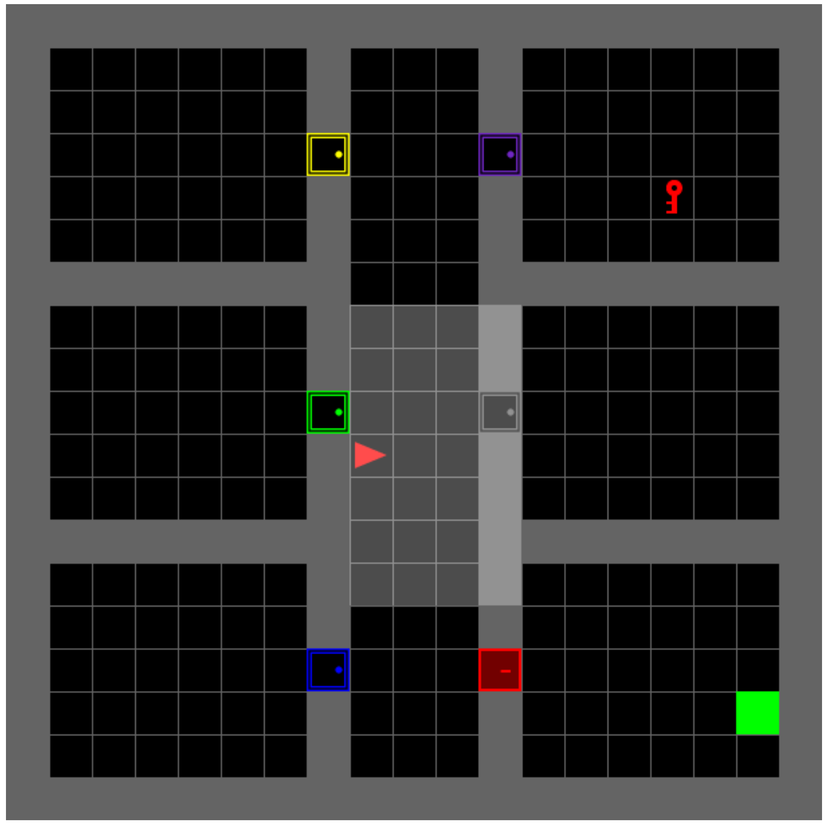
MuJoCo Environments

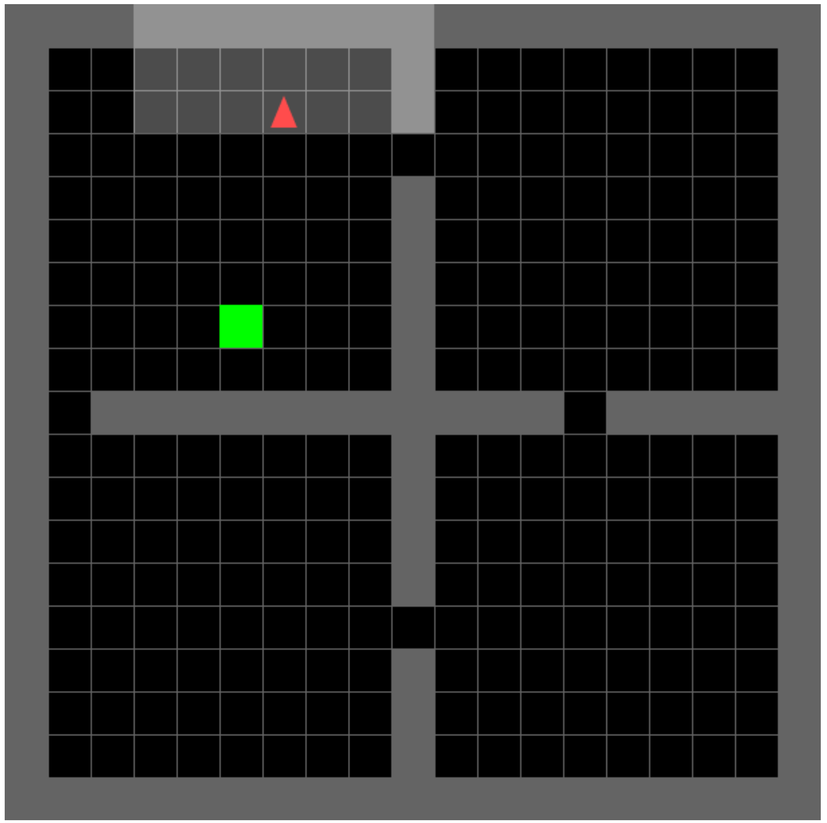







FrankaKitchen



MiniGrid





Minari

- The best hub for Offline Reinforcement Learning datasets
- Released by the Farama Foundation (same maintainers as the Gymnasium environments)
- Makes creating and uploading datasets really easy.

```
(venv) bryce@brycelab2:~/Repos/offline-gp$ minari --help

 Usage: minari [OPTIONS] COMMAND [ARGS]...

 Minari is a tool for collecting and hosting Offline datasets for Reinforcement
 Learning environments based on the Gymnaisum API.

╭─ Options ──────────────────────────────────────────────────────────────────────╮
│ --version                -v      Show installed Minari version.                │
│ --install-completion             Install completion for the current shell.     │
│ --show-completion                Show completion for the current shell, to     │
│                                  copy it or customize the installation.        │
│ --help                           Show this message and exit.                   │
╰────────────────────────────────────────────────────────────────────────────────╯
╭─ Commands ─────────────────────────────────────────────────────────────────────╮
│ list       List Minari datasets in local or remote storage.                    │
│ show       Describe a local or remote dataset, and its environment.            │
│ delete     Delete datasets from local database.                                │
│ download   Download Minari datasets from Farama server.                        │
│ upload     Upload Minari datasets to the remote Farama server.                 │
│ combine    Combine multiple datasets into a single Minari dataset.             │
╰────────────────────────────────────────────────────────────────────────────────╯
```

Name	# Episodes	# Steps	Size	Author
mujoco/... *Group with 28 datasets*	54K	23M	24.26 GB	Kallinteris Andreas
minigrid/... *Group with 170 datasets*	170K	4M	7.51 GB	Omar G. Younis
metaworld/... *Group with 50 datasets*	2K	1M	474.9 MB	Omar G. Younis
webagents/weblinx-v0	969	24K	135.1 MB	Zdeněk Kasner, Siva Reddy, Xing Han Lù
D4RL/antmaze/medium-play-v1	1K	1M	605.2 MB	Alex Davey
D4RL/antmaze/umaze-diverse-v1	1K	1M	605.0 MB	Alex Davey
D4RL/antmaze/large-diverse-v1	1K	1M	605.2 MB	Alex Davey
D4RL/antmaze/large-play-v1	1K	1M	605.2 MB	Alex Davey
D4RL/antmaze/medium-diverse-v1	1K	1M	605.2 MB	Alex Davey
D4RL/antmaze/umaze-v1	1K	1M	605.0 MB	Alex Davey
D4RL/pointmaze/open-dense-v2	10K	1M	437.6 MB	Rodrigo Perez-Vicente
D4RL/pointmaze/umaze-v2	13K	1M	556.2 MB	Rodrigo Perez-Vicente
D4RL/pointmaze/large-dense-v2	3K	1M	239.2 MB	Rodrigo Perez-Vicente
D4RL/pointmaze/medium-v2	5K	1M	284.0 MB	Rodrigo Perez-Vicente
D4RL/pointmaze/umaze-dense-v2	13K	1M	556.2 MB	Rodrigo Perez-Vicente
D4RL/pointmaze/medium-dense-v2	5K	1M	284.0 MB	Rodrigo Perez-Vicente

Key Takeaways

- Offline RL is hard because we can't explore and we can't rely on feedback to correct bad behaviours
- The biggest issue is distributional shift: small errors compounding.
- Policy constraints, conservative value estimates, imitation learning are techniques to mitigate OOD actions.
- Importance sampling can be used to weight points based on their likelihood of relevancy to the new policy.
- Lots of diverse solutions: model-based methods, sequence modelling, **evolutionary algorithms.**

References

Fitted Q-Iteration

Ernst, D., Geurts, P., & Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, *6*, 503–556.

BCQ

S. Fujimoto, D. Meger, and D. Precup, “Off-policy deep reinforcement learning without exploration,” in ICML, 2019, pp. 2052–2062.

BEAR

A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, “Stabilizing off-policy Q-learning via bootstrapping error reduction,” in Proc. NeurIPS, 2019, pp. 1–11

BRAC

Y. Wu, G. Tucker, and O. Nachum, “Behavior regularized offline reinforcement learning,” 2019, arXiv:1911.11361.

References

AWR

X. B. Peng, A. Kumar, G. Zhang, and S. Levine, “Advantage-weighted regression: Simple and scalable off-policy reinforcement learning,” CoRR, vol. abs/1910.00177, pp. 1–19, Oct. 2019.

AWAC

A. Nair, A. Gupta, M. Dalal, and S. Levine, “AWAC: Accelerating online reinforcement learning with offline datasets,” 2020, arXiv:2006.09359.

References

TD3+BC

S. Fujimoto and S. S. Gu, “A minimalist approach to offline reinforcement learning,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 20132–20145.

TD3

S. Fujimoto, H. Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in Proc. ICML, 2018, pp. 1582–1591.

DDPG

T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous control with deep reinforcement learning," in Proc. ICLR, 2016.

References

CQL

A. Kumar, A. Zhou, G. Tucker, and S. Levine, “Conservative Q-learning for offline reinforcement learning,” in Proc. NeurIPS

REM

R. Agarwal, D. Schuurmans, and M. Norouzi, “An optimistic perspective on offline reinforcement learning,” in Proc. ICML, 2020, pp. 104–114.