

ECE7121 Learning-based control – 2025 Fall

Offline RL



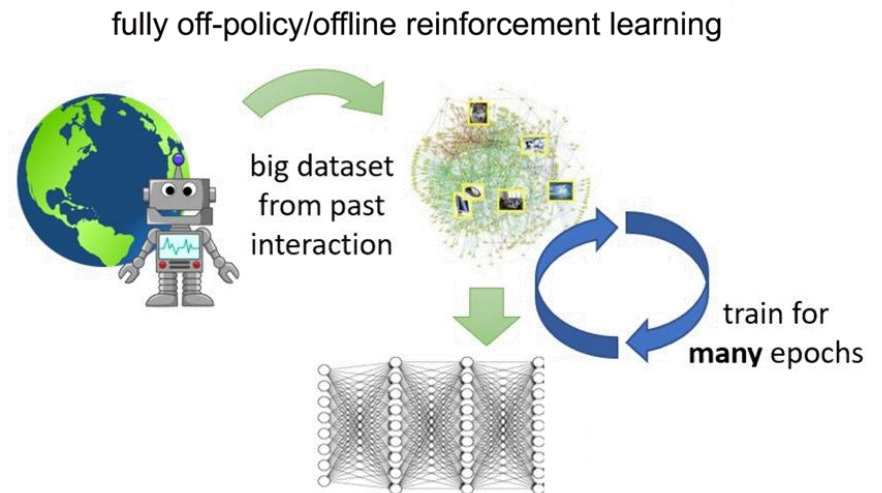
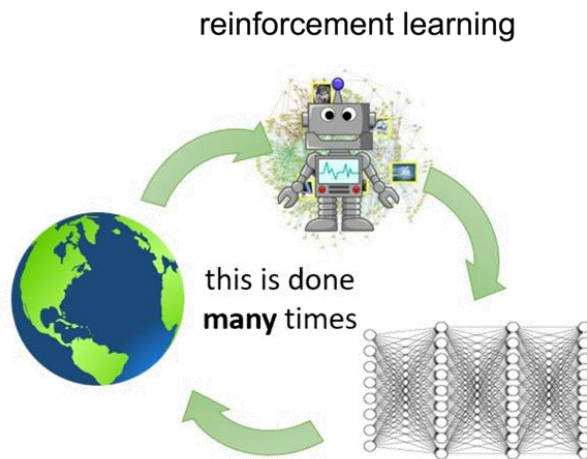
INHA UNIVERSITY

Overview

- > Offline RL
 - Motivation, OOD over-estimation
 - off-policy evaluation
- > Model-free offline RL
 - direct constraint
 - indirect constraint
- > Model-based offline RL
 - MOPO, COMBO
- > Sequence generation model
 - DT
 - TT
 - Diffuser

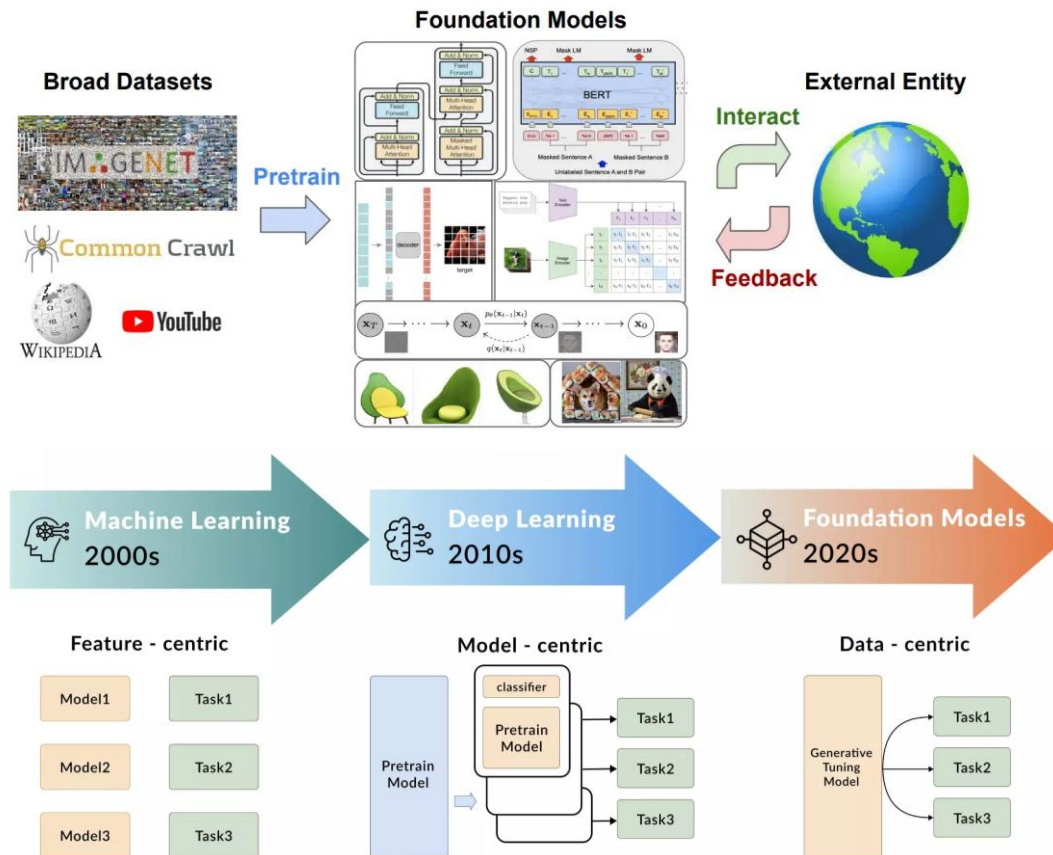
Introduction

- > Online RL is unable to collect lots of data
 - rely exclusively on interactions
 - many interactions can't be done in reality
 - drug discovery experiment
 - robot control, autonomous driving (collision avoidance)
 - Solution
 - incorporate existing, static datasets



Introduction

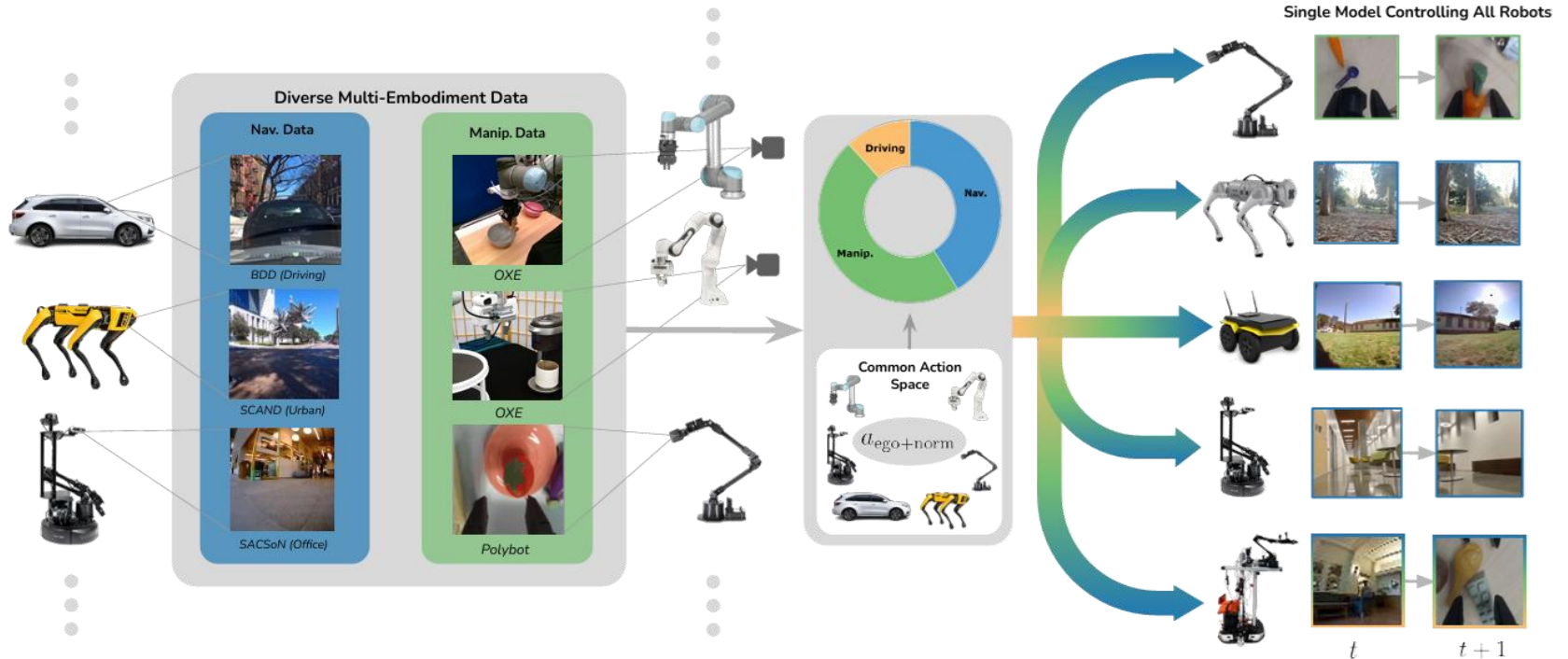
- > Machine learning trends
 - high-level intelligence (diverse tasks, generalization)
 - require big data and big models (foundation model)



Introduction

> Can we do that in robot control tasks?

- cross-embodiment learning (2024)
- manipulation, driving, and navigation

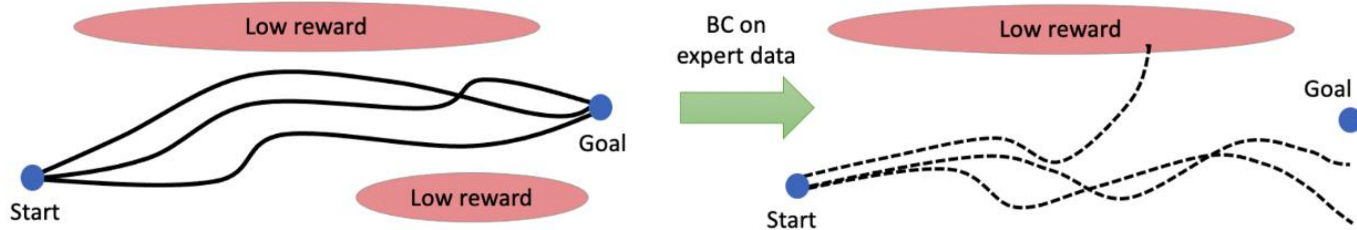


Introduction

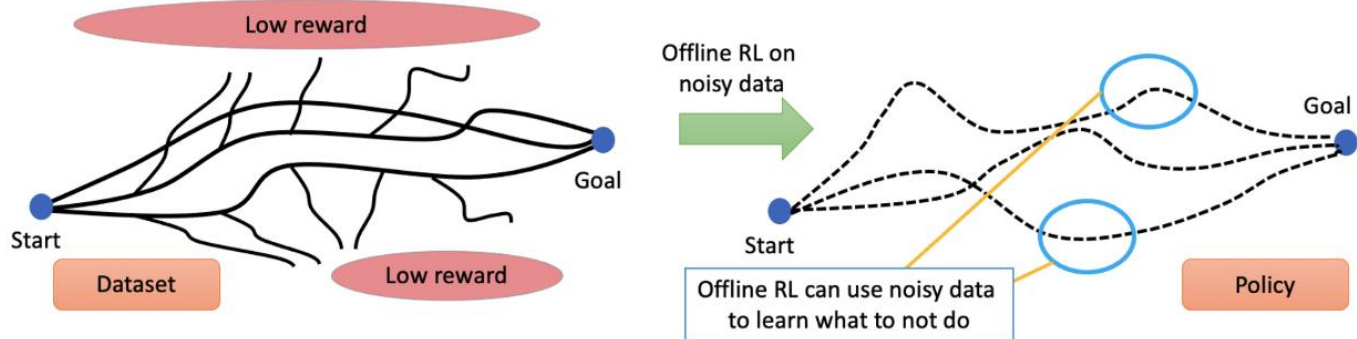
> Offline RL process

- leverage datasets collected by people, existing systems
- different with the imitation learning!
 - data collection policy doesn't need to be expert

Behavior cloning on expert:



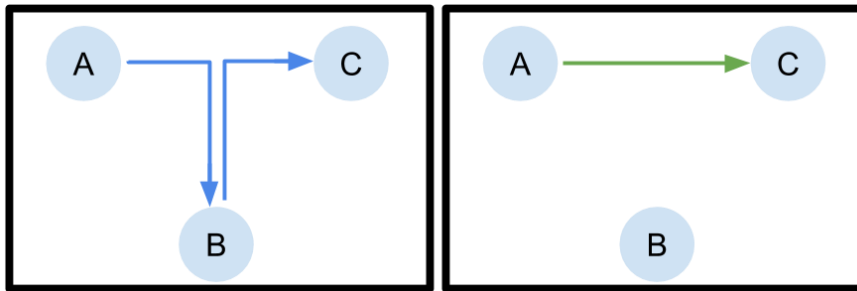
Offline RL with noisy data:



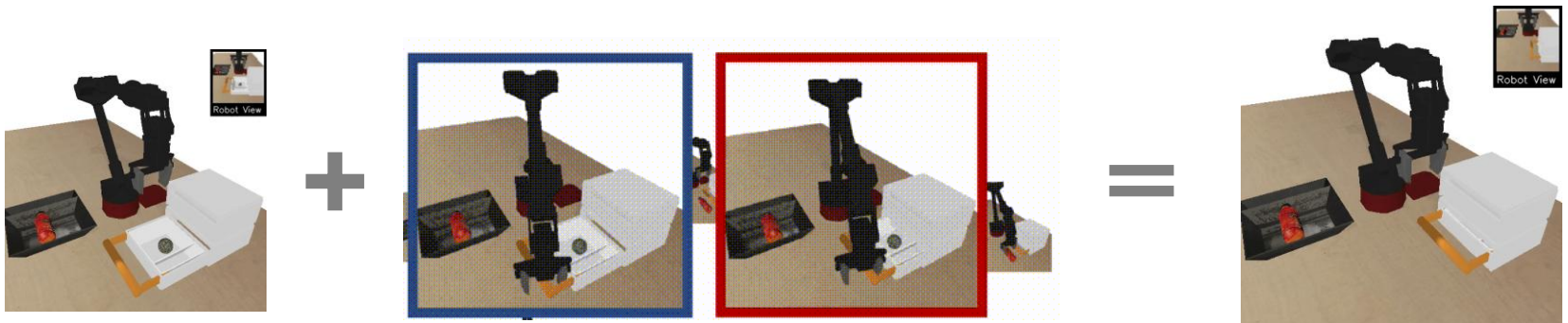
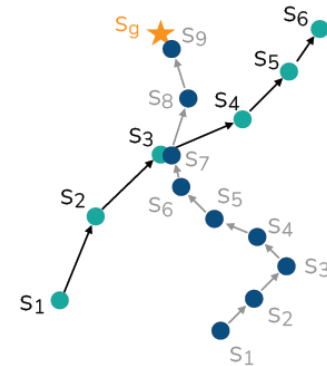
Introduction

> How can it work?

- find the good stuff in a dataset full of good and bad behaviors
- parts of good behaviors can be recombined



- example) connecting skills



Recall: off-policy learning

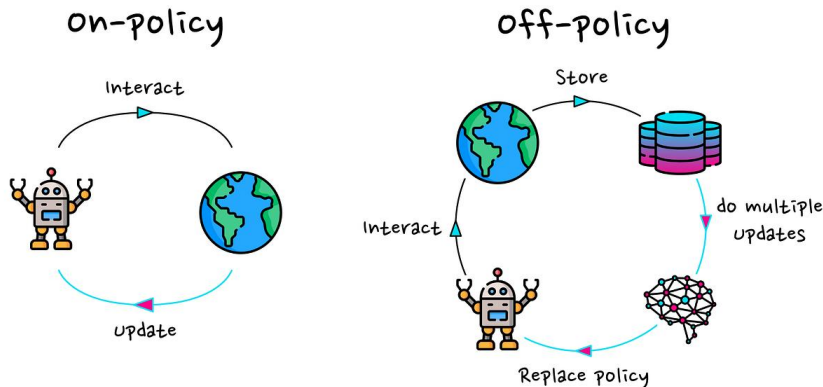
> Q-learning (off-policy TD learning)

- $Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_a Q(s', a') - Q(s, a))$

> Off-policy AC

- Version 2. Replay buffer

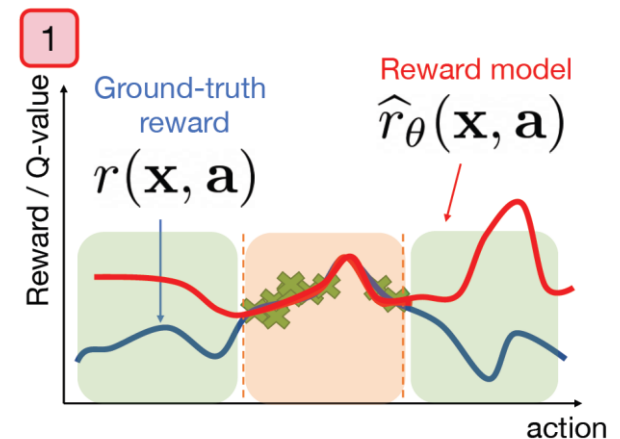
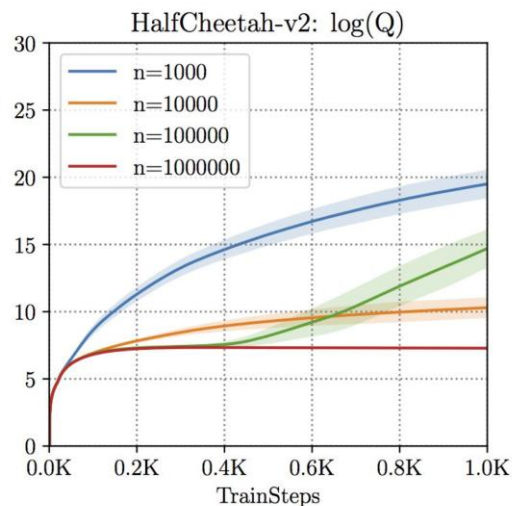
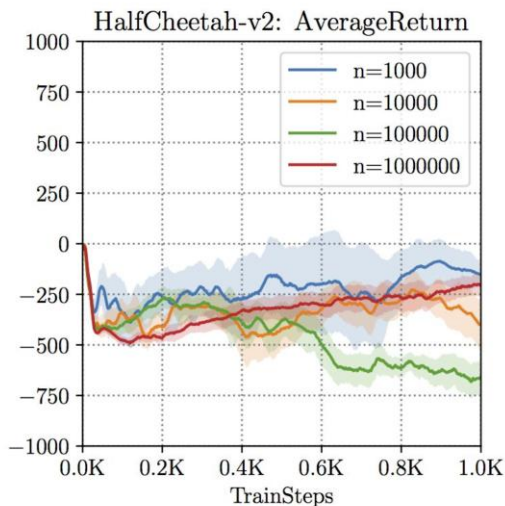
1. collect experience $\{s_i, a_i\}$ from π_ϕ and add to replay buffer
2. sample a batch $\{s_i, a_i, r_i, s_i'\}$ from buffer \mathcal{R}
3. update \hat{Q}_θ^π using targets $r(s_i, a_i) + \gamma \hat{Q}_\theta^\pi(s_i', a_i')$ for each s_i, a_i
4. $\nabla_\phi J(\phi) \approx \sum_{i=1}^N \nabla_\phi \log \pi_\phi(a_i | s_i) \hat{Q}_\theta^{\pi_\phi}(s_i, a_i)$ where $a_i \sim \pi_\phi(a | s_i)$
5. $\phi \leftarrow \phi + \alpha \nabla_\phi J(\phi)$



Off-policy learning

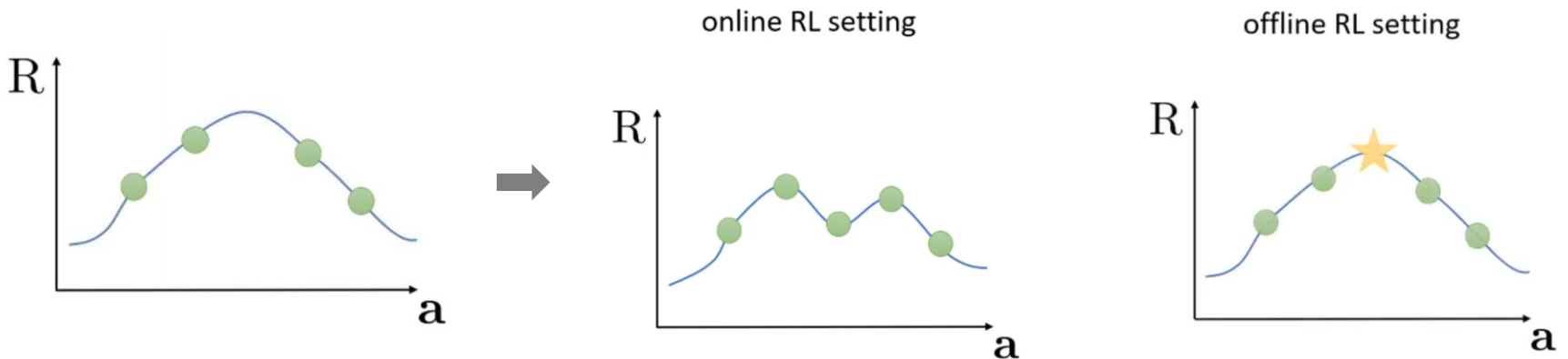
> Employing off-policy RL doesn't work

- we learn value function \hat{Q}_θ^π , but it cannot predict the results for unseen actions
 - maximizing over unseen actions introduces bootstrapping error
 - these errors propagate without correction, Q-value can diverge
- policy will seek out actions where Q -function is over-optimistic



Off-policy learning

- > Online RL algorithms don't have to handle this because they can simply try this action and see what happens
- > Offline RL methods must somehow account for these unseen out-of-distribution (OOD) actions, ideally in a safe way



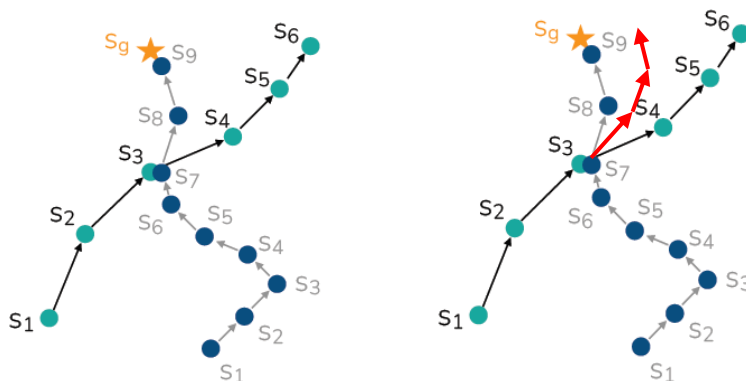
Difficulty of offline RL

> Exploration is impossible

- we assume that dataset covers the space of high-reward transitions to make learning feasible
- Yet, there are no known non-trivial sufficiency conditions on datasets

> Counterfactual queries (what if questions)

- what might happen if the agent were to carry out a new action
- make the learned policy to perform better than the observed behaviors
- In standard learning, we assume i.i.d, but here we need to learn differently
- In behavior cloning under a strong assumption, error bound is at best quadratic in the time horizon in the offline setting, but linear in the online



once it encounters OOD states,
the errors keeps accumulating

Offline evaluation

> Off-policy evaluation via importance sampling

- recall: importance sampling $\mathbb{E}_{x \sim p(x)}[f(x)] = \mathbb{E}_{x \sim q(x)} \left[\frac{p(x)}{q(x)} f(x) \right]$

- RL objective: maximize return

$$J(\pi) = \mathbb{E}_{\tau \sim p_{\pi}(\tau)} [\sum_{t=0}^H \gamma^t r(s_t, a_t)]$$

- evaluate the $J(\pi)$ with trajectories sampled from $\pi_{\beta}(\tau)$ (data collection)

$$\begin{aligned} J(\pi_{\theta}) &= \mathbb{E}_{\tau \sim \pi_{\beta}(\tau)} \left[\frac{\pi_{\theta}(\tau)}{\pi_{\beta}(\tau)} \sum_{t=0}^H \gamma^t r(s_t, a_t) \right] \\ &\approx \frac{1}{n} \sum_{i=1}^n \sum_{t=0}^H w_t^i \gamma^t r_t^i \end{aligned}$$

$$\text{where } w_t^i = \frac{1}{n} \prod_{t'=0}^t \frac{\pi_{\theta}(a_{t'}^i | s_{t'})}{\pi_{\beta}(a_{t'}^i | s_{t'})}$$

- The estimator can have high variance
- To reduce the variance, we may use the doubly robust estimator

$$J(\pi_{\theta}) \approx \sum_{i=1}^n \sum_{t=0}^H \gamma^t \left(w_t^i \left(r_t^i - \hat{Q}^{\pi_{\theta}}(s_t, a_t) \right) - w_{t-1}^i \mathbb{E}_{a \sim \pi_{\theta}}[a | s_t] [\hat{Q}^{\pi_{\theta}}(s_t, a)] \right)$$


Offline evaluation

> Estimation of value for a given policy leads to policy optimization

- policy gradient

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim p_{\pi_{\theta}}(\tau)} \left[\sum_{t=0}^H \gamma^t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \underbrace{\left(\sum_{t'=t}^H \gamma^{t'-t} r(s_{t'}, a_{t'}) - b(s_t) \right)} \right]$$

- use importance sampling as well

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi_{\beta}(\tau)} \left[\frac{\pi_{\theta}(\tau)}{\pi_{\beta}(\tau)} \sum_{t=0}^H \gamma^t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \hat{A}(s_t, a_t) \right]$$


- To reduce the variance, we may use a regularizer over importance weights
- The importance-weighted objective requires multiplying per-action importance weights over the time steps, which leads to very high variance

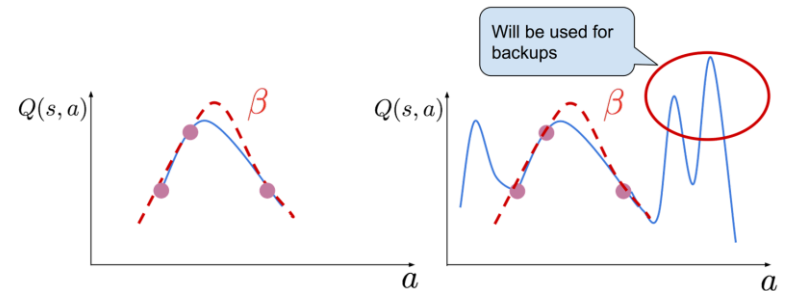
Offline evaluation

- > There are some more techniques to regularize the optimization
 - handling the bias and variance of the objective function
 - e.g., approximate off-policy policy gradient, marginalized importance sampling
- > Off-policy evaluation through importance sampling have been used for a classic off-policy learning
 - applying such methods in the fully offline setting poses many challenges
 - when the behavior policy is too different from the current learned policy, the importance weights will become degenerate
 - any estimate of the return or the gradient will have too much high variance
 - it is most suitable in the case where the policy only deviates by a little

Constraining policy

- > The problem occurs from unseen actions (distribution shift)
 - the agent picks actions that are poorly covered by the dataset
 - constraint allows useful novelty while preventing harmful novelty
 - the goal is to learn a policy close to the behavior policy

$$\pi_{\theta} \approx \pi_{\beta}$$



> Approaches

- explicit constraint
 - directly constrains the policy so that it remains close to the behavior policy
- implicit constraints
 - constraint the value function or occupancy that indirectly induces a policy similar to the behavior policy
- uncertainty-based methods
 - estimate the uncertainty of action outcomes
 - avoid actions with high uncertainty (unseen/unpredictable actions)

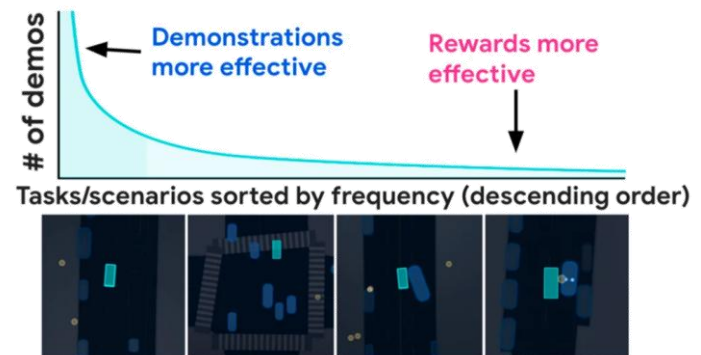
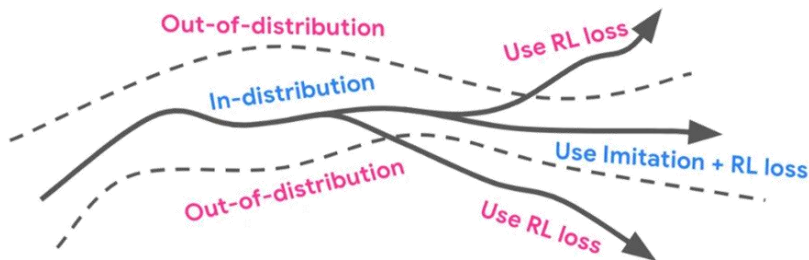
Direct constraints on the policy

> Behavior-cloning regularization – TD3+BC (2021)

- TD3 policy objective $J(\theta) = \mathbb{E}_{(s,a) \sim \mathcal{D}}[Q(s, \pi(s))]$
- maximize Q but tether the actor to dataset actions

$$\max \mathbb{E}_{(s,a) \sim \mathcal{D}}[\lambda Q(s, \pi(s)) - (\pi_{\theta}(s) - a)^2]$$

- Overall good performance
- simple baseline with demonstrations are decent (above medium)
- works on real self-driving problems (Waymo, 2023)



Direct constraints on the policy

- > Advantage-weighted imitation – AWAC (2020) / CRR (2020), IQL(2021)
 - imitate the dataset, but up-weight actions the critic deems above-average
 - BC: just imitate the policy

$$\max \mathbb{E}_{(s,a) \sim \mathcal{D}} [\log \pi(a|s)]$$

- AWI: use advantage $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$ (larger for good action)

$$\max \mathbb{E}_{(s,a) \sim \mathcal{D}} [w(A^\pi(s, a)) \log \pi(a|s)]$$

- AWAC: exponential weight, CRR: binary clipping
- Then, project the learned policy $\bar{\pi}$ onto the policy class

$$\pi_{k+1} \leftarrow \arg \min_{\pi} D_{KL}[\bar{\pi}_{k+1} \parallel \pi]$$

Direct constraints on the policy

> Divergence to behavior constraints – BRAC (2019) family

- explicitly limit how far π_θ can move from the behavior π_β

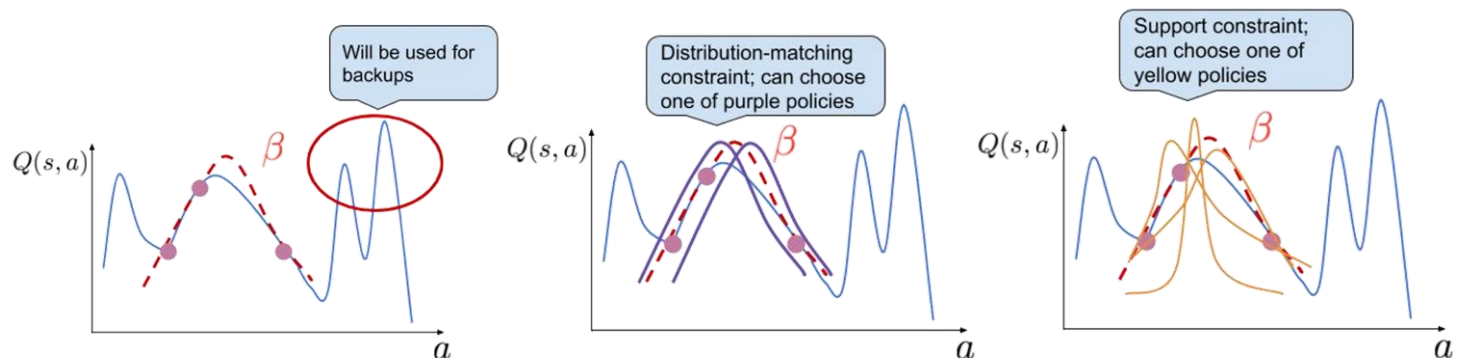
$$\max \mathbb{E}_{(s,a) \sim \mathcal{D}} \left[Q(s, \pi(s)) - \alpha D[\pi(s) \parallel \pi_\beta(s)] \right]$$

or $\max \mathbb{E}_{(s,a) \sim \mathcal{D}} [Q(s, \pi(s))] \quad \text{s.t.} \quad D[\pi(s) \parallel \pi_\beta(s)] \leq \epsilon$

- D is a distance measure between two policy distributions (e.g., KL divergence, total variation distance, Pearson divergence)
- easy to implement but not necessarily what we want
- Support constraint: $\pi(a|s) \geq 0$ only if $\pi_\beta(a|s) \geq \epsilon$ (BEAR, 2019)
 - more complex to implement but much closer to what we really want
 - maximum mean discrepancy resembles a support constraining metric
- Generally, the best modern offline RL methods do not use these

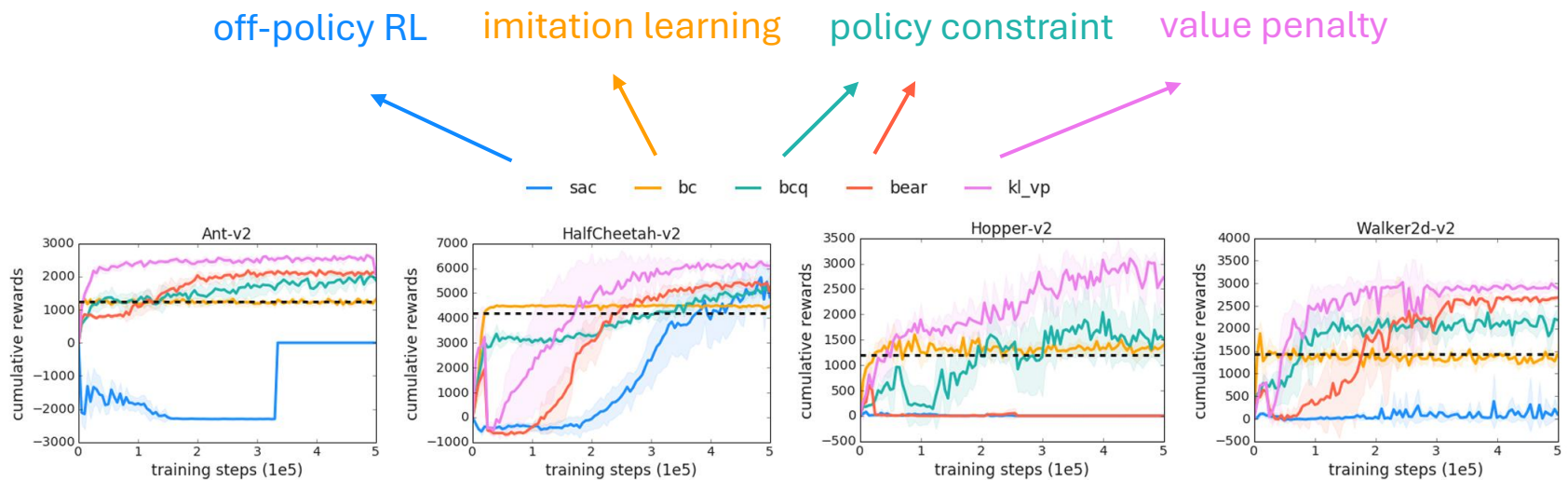
Direct constraints on the policy

- > Support-restricted policy - BCQ, SPIBB
 - Disallow (or strongly down-weight) actions not sufficiently present
 - BCQ (batch-constrained Q-learning, 2019)
 - trains a generative model of the behavior policy
 - sample a set of candidates from this model (the policy never selects actions outside the dataset's support)
 - SPIBB (safe policy improvement with baseline bootstrapping, 2019)
 - define a support set of sufficiently frequent actions in the dataset
 - the learned policy can deviate only within this support set
 - for infrequent actions, the policy is forced to follow the behavior policy



Indirect constraints on the policy

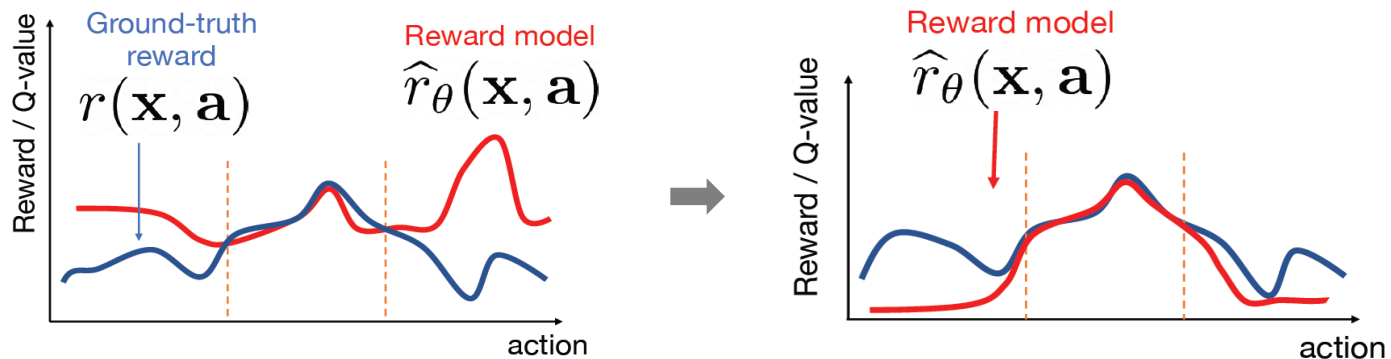
- > Instead of constraining the policy distribution directly, impose constraints on related quantities (e.g., value function, occupancy)
 - the policy ends up avoiding out-of-support actions
 - often more practical when the behavior policy is hard to model



- value penalty outperforms policy regularization

Indirect constraints on the policy

- > Conservative Q-learning (CQL, 2020)
 - make Q-values pessimistic for unseen actions



$$\begin{aligned}
 & \text{always pushes Q-values down} \quad \quad \quad \text{push up on } (\mathbf{s}, \mathbf{a}) \text{ samples in data} \\
 & \quad \quad \quad \downarrow \quad \quad \quad \downarrow \\
 & \hat{Q}^\pi = \arg \min_Q \max_\mu \left\{ \begin{aligned} & \alpha E_{\mathbf{s} \sim D, \mathbf{a} \sim \mu(\mathbf{a}|\mathbf{s})} [Q(\mathbf{s}, \mathbf{a})] - \alpha E_{(\mathbf{s}, \mathbf{a}) \sim D} [Q(\mathbf{s}, \mathbf{a})] \\ & + E_{(\mathbf{s}, \mathbf{a}, \mathbf{s}') \sim D} \left[(Q(\mathbf{s}, \mathbf{a}) - (r(\mathbf{s}, \mathbf{a}) + E_\pi [Q(\mathbf{s}', \mathbf{a}')]))^2 \right] \end{aligned} \right\} \mathcal{L}_{\text{CQL}}(\hat{Q}^\pi)
 \end{aligned}$$

- CQL is one way to construct a conservative Q value
- Generalized conservative Q-value: ATACL, 2022

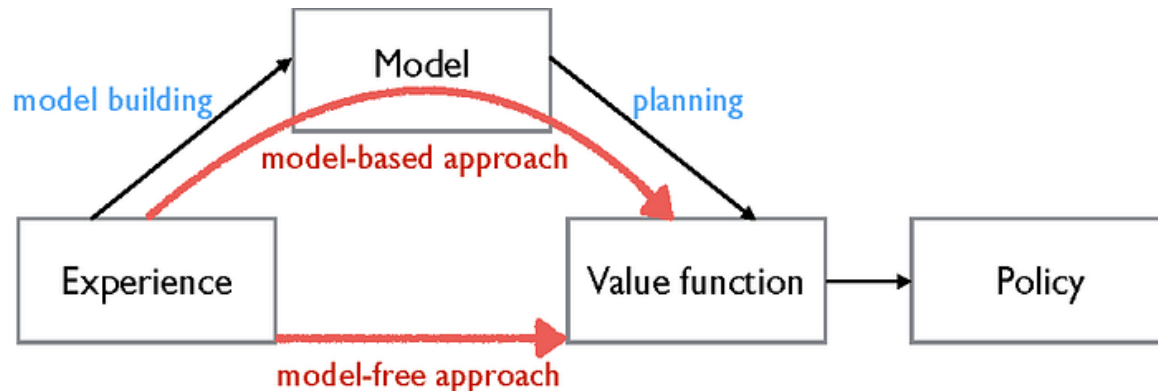
Constraints on the policy

> Note

- uncertainty-based methods are conceptually attractive, but it is very hard to obtain calibrated uncertainty
 - good measure for exploration, but not for trustworthiness
- policy constraint methods suffer from several challenges
 - estimation of behavior policy is hard (highly multimodal behaviors)
 - when the size of dataset is limited, approximate DP tends to overfit
- methods that estimate a conservative or pessimistic value function present a somewhat different set of tradeoffs
 - they avoid issues of estimating the behavior policy
 - excessive pessimism becomes the bigger issue (under-estimate for under-sampled states)
 - how to dynamically modulate the degree of conservatism

Offline model-based RL

- > MBRL: leverage a learned (dynamics) model to plan or augment data, improving sample-efficiency
 - if there is a model error (due to distribution shift), the policy may exploit it
 - core idea: add pessimism or robustness into planning



Offline model-based RL

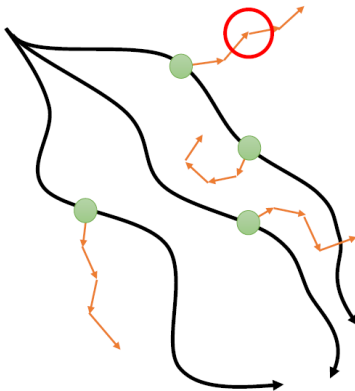
> Uncertainty-penalized reward (MOPO, 2020)

- plan cautiously in the model
- replace the model reward with a pessimistic reward

$$r'(s, a) = r(s, a) - \lambda u(s, a)$$

where $u(s, a)$ is an uncertainty penalty

- uncertainty is measured by ensemble disagreement
- use short rollout horizon to prevent compounding model error



Offline model-based RL

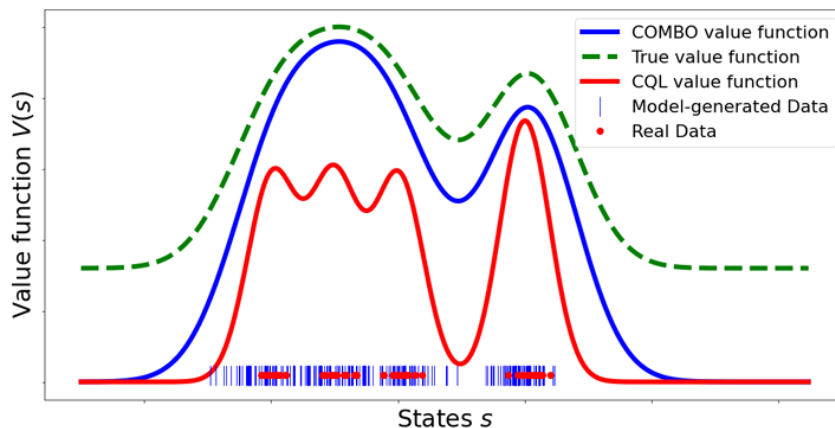
> Conservative value learning (COMBO, 2021)

- Algorithm

- learn a probabilistic dynamics model
- simulate a model to generate an artificial dataset
- conservative value learning with mixed dataset

- Why it works

- model rollouts expose where the current policy will actually go
- actor prefers in-support behavior while still gaining coverage from model data



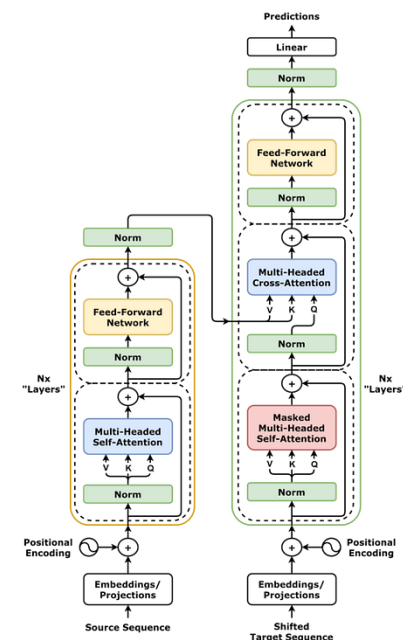
Offline model-based RL

> Note

- standard off-policy MBRL (such as MBPO) works reasonably on offline RL
- uncertainty estimation for models is in some ways more straightforward than uncertainty for value functions, yet it leaves much to be desired
- some MDPs are easy to model; others are exceedingly difficult
- it is still an open question whether model-based RL can improve over model-free DP
 - DP also essentially uses the dataset as a non-parametric model
 - In the linear function approximation case, model-based updates and fitted value iteration updates produce identical iterates (yet, unknown for non-linear function approximation)

Transformer

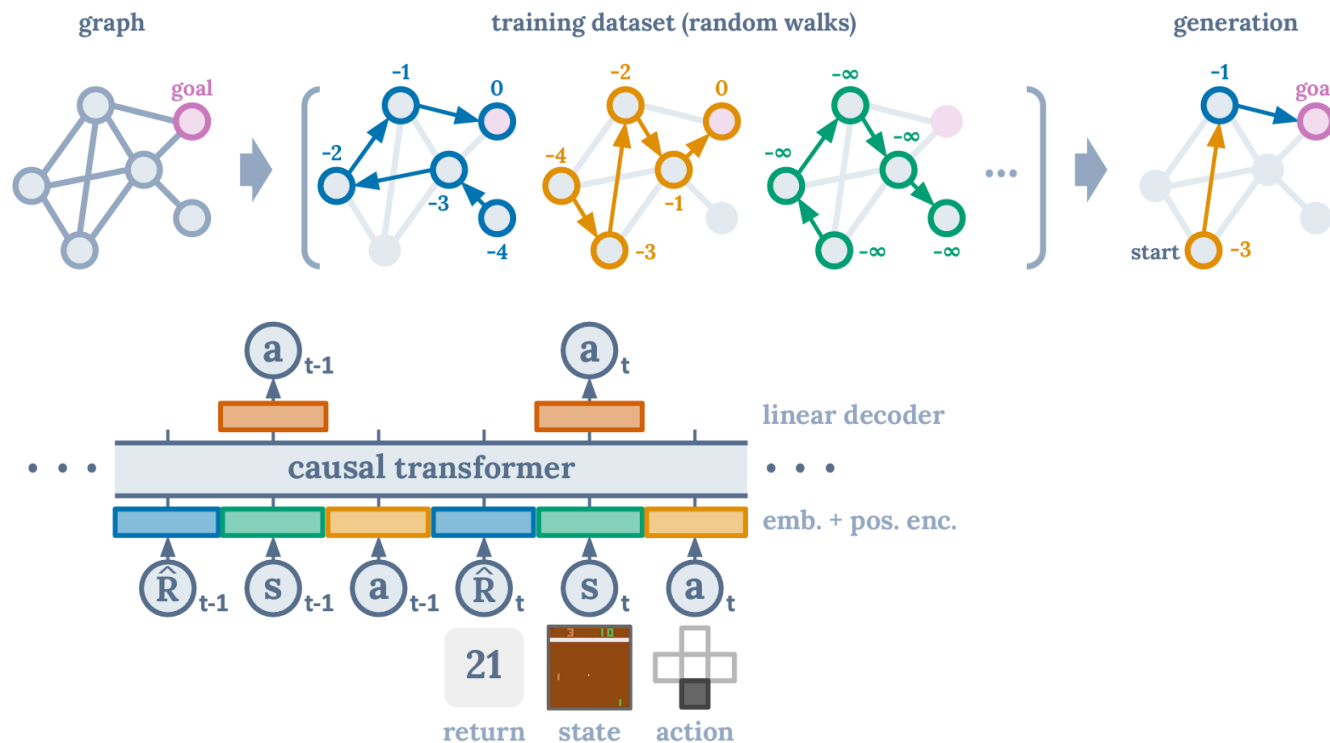
- > Sequence model using attention, enables long-term dependency
 - Key component: positional encoding provides order information, multi-head attention captures multiple relations simultaneously
 - it enables the model to capture relationships between all elements in a sequence, regardless of their distance
 - unlike RNN, transformers process entire sequences in parallel
 - transformer can handle very long sequences and large datasets
 - challenges:
 - computational cost
 - data requirements
 - interpretability



Transformer

> Decision transformer (2021)

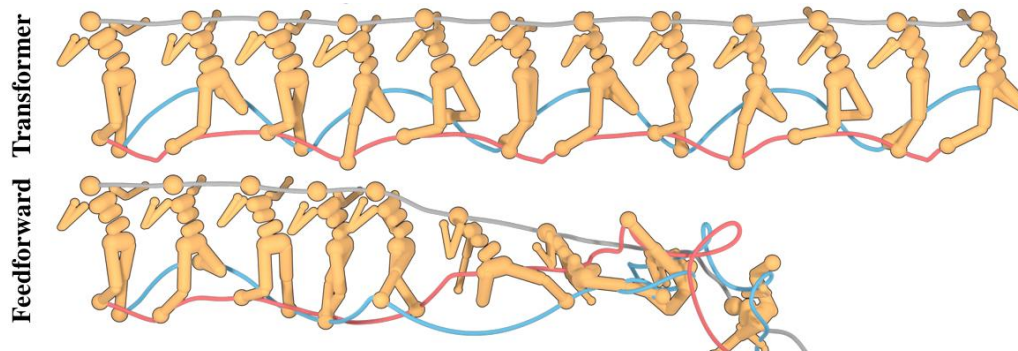
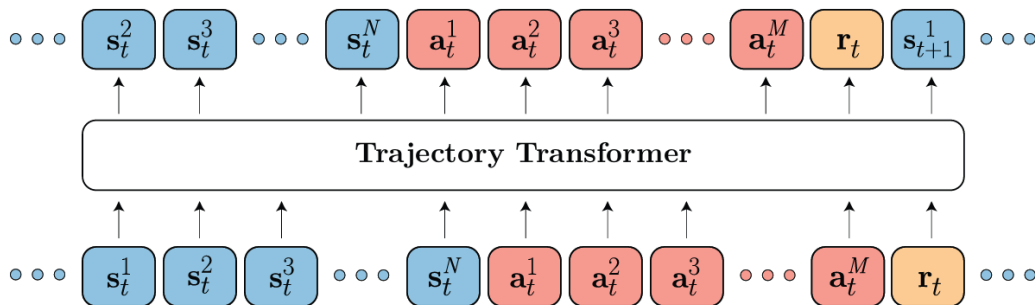
- reframe RL as a conditional sequence modeling problem (RL→SL)
- treat trajectories as sequences and predict the next action autoregressively
- return-to-go is used as a conditioning signal



Transformer

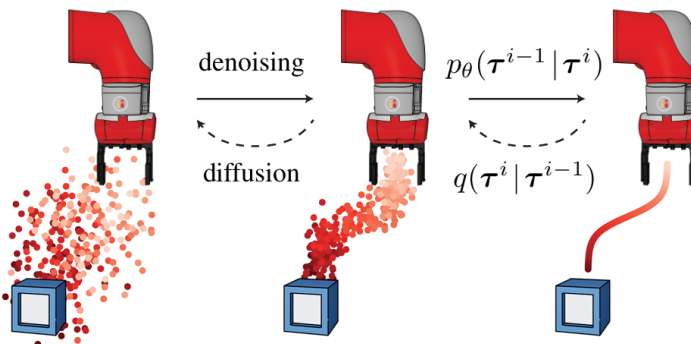
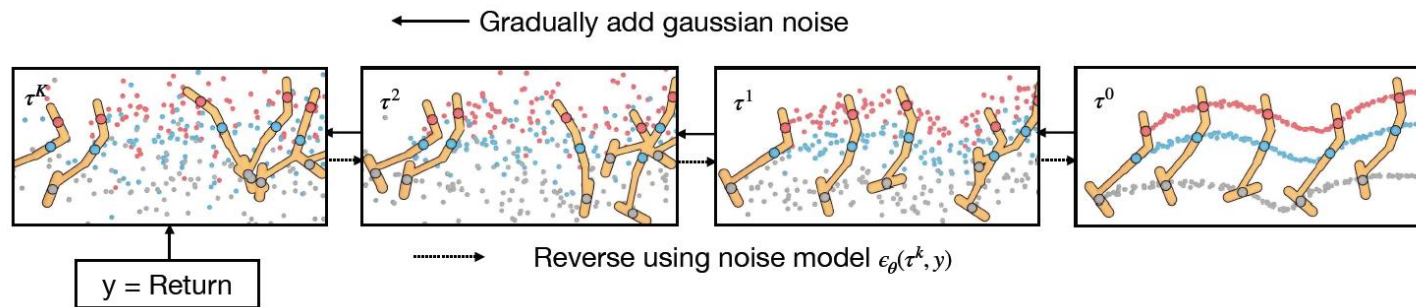
> Trajectory transformer (2021)

- instead of predicting action, predict the trajectory (state, action, reward)
- discretize state and action dimensions into tokens
- use planning (beam search) over predicted trajectories
 - keeps only the top-B partial sequences (beams)



Diffuser

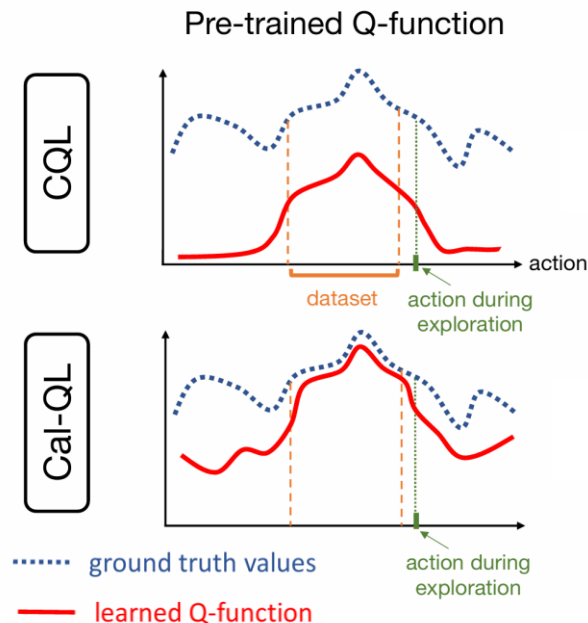
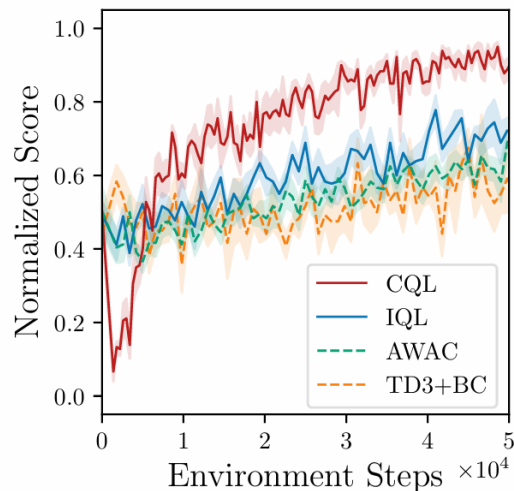
- > Learning a conditional trajectory model with diffusion
 - forward noising: add Gaussian noise to entire trajectories
 - reverse denoising model: recover clean trajectories
 - Generative prior: produces trajectories with strong long-horizon coherence
 - Conditioning: bias sampling toward goals, rewards, or constraints



Extensions

> Online-finetuning

- offline RL allows us to train on all existing data to learn policy initializations that can be improved online
- CQL suffers from initial policy unlearning
 - CQL learns conservative Q values \rightarrow magnitudes are much smaller
- Calibrated-QL: never push down Q-values if they are smaller than a reference Q-value



Summary

- > Which offline RL algorithm to use?
 - if you want to only train offline
 - conservative Q-learning: just one hyperparameter, widely tested
 - If you want to train offline and finetune online
 - advantage-weighted actor-critic (AWAC): widely used and well tested
 - implicit Q-learning: more flexible, more hyperparameters
 - If you have a good way to train models in your domain
 - COMBO: similar properties as CQL, but benefits from models
 - trajectory transformer: very powerful models, expensive computation

Summary

> Future of offline RL

- In CV/NLP, progress has come from datasets as much as methods
- Large, diverse, representative datasets often more impactful than new algorithms
- In RL, active data collection is costly, impractical, and unsafe
- Offline RL reframes RL as counterfactual inference
- Data-driven RL has the potential to open a new era of real-world applications

Reference

> Reference

- <https://arxiv.org/pdf/2005.01643>
- <https://bair.berkeley.edu/blog/2019/12/05/bear/>