MOPO: Model-based Offline Policy Optimization

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou1, Sergey Levine, Chelsea Finn, Tengyu Ma

Claas A Voelcker

STA 4273 - University of Toronto

Summary

- The title says it all
    - Model-based: learning environment models
    - Offline: learning with offline (precollected) data
    - Policy Optimization: learning a policy
- Review the background
- Review the theory
- Review the algorithm
- Poke at the weak spots

Introduction

The challenge: Offline data

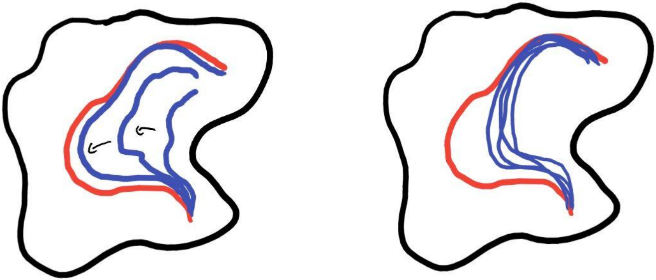


Figure 1: A crude visualization of offline RL (left online, right offline)

The challenge: Offline data

Challenges:

- Might not contain correct solution
- Intermediate policies could lead outside of data covered region
- Generalization of RL algorithms unclear

Solutions:

- Inverse reinforcement learning
- Regularization towards data distribution
- Hope for generalization
- Model-based RL?

An answer? Model learning

Why model learning?

- Supervised: more hopes of generalization
- Model can cover region of low data
- We can estimate model uncertainty

Classic algorithm: Dyna

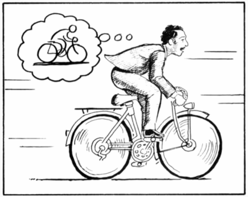


Figure 2: Comic from Ha, Schmidhuber: ”World Models”, (`https://arxiv.org/pdf/1803.10122.pdf`)

Dyna based RL

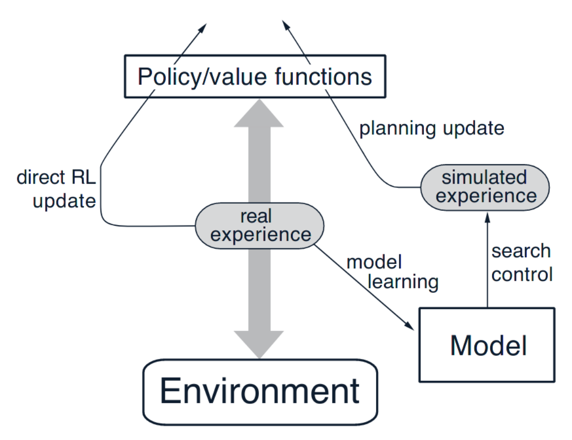


Figure 3: Diagram from Sutton, Barto: "Reinforcement Learning: An Introduction", p.163, MIT Press 2018

MBPO as offline-learning

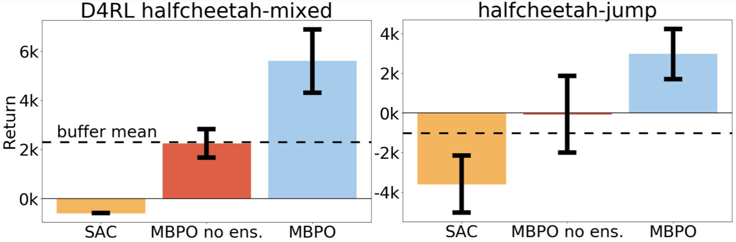


Figure 4: Comparison of previous methods on offline benchmarks, diagram from paper

Offline optimized model-based RL

When to trust your model (revisited)

Disclaimer: Compressed notation for intuition, not rigorous

Try to quantify the error when executing policies π from one model $\hat{T}$ in another T

Expected discounted return :

$$\eta_T(\pi) := \mathbb{E}_T \left[\sum \gamma^t r(s_t, a_t) \right]$$

Difference in value function :

$$G^\pi(s,a) := \mathbb{E}_{s' \sim \hat{T}(s,a)}[V_T^\pi(s')] - \mathbb{E}_{s' \sim T(s,a)}[V_T^\pi(s')]$$

How good are we sure to be?

Estimate expected return under true dynamics T

$$\begin{aligned}
\eta_{\hat{T}}(\pi) - \eta_T(\pi) =& \gamma \mathbb{E}^{\pi}_{\hat{T}} \left[\sum \gamma^t G^{\pi}_{\hat{T}}(s_t, a_t) \right] \\
\eta_T(\pi) =& \mathbb{E}^{\pi}_{\hat{T}} \left[\sum \gamma^t \left(r(s_t, a_t) - \gamma G^{\pi}_{\hat{T}}(s_t, a_t) \right) \right] \\
\geq& \mathbb{E}^{\pi}_{\hat{T}} \left[\sum \gamma^t \left(r(s_t, a_t) - \gamma |G^{\pi}_{\hat{T}}(s_t, a_t)| \right) \right]
\end{aligned}$$

Need $|G^{\pi}(s, a)| = |\mathbb{E}_{s' \sim \hat{T}(s,a)}[V^{\pi}(s')] - \mathbb{E}_{s' \sim T(s,a)}[V^{\pi}(s')]|$

$$|G^{\pi}_{\hat{T}}(s, a)| \leq \sup_{V \in \mathcal{F}} \left| \mathbb{E}_{s' \sim T}[V(s'|s, a)] - \mathbb{E}_{s' \sim \hat{T}}[V(s'|s, a)] \right| = d_{\mathcal{F}}(\hat{T}(s, a), T(s, a))$$

- For $\mathcal{F}$ bounded: Total variation distance
- For $\mathcal{F}$ Lipschitz-smooth: Wasserstein distance

How good are we sure to be?

Idea: expected return in T is lower bounded by:

$$\mathbb{E}_{\hat{T}}\left[\sum \gamma^t \left(r(s_t, a_t) - \gamma d_{\mathcal{F}}(\hat{T}(s_t, a_t), T(s_t, a_t))\right)\right] \tag{1}$$

- new MDP with $\tilde{r}(s, a) = r(s, a) - \gamma d_{\mathcal{F}}(\hat{T}(s, a), T(s, a))$
- optimize policy here
- by previous, return will underestimate true return (achieve conservative learning)

Big problem: don't know T and therefore also not $d_{\mathcal{F}}(T(\hat{s, a}), T(s, a))$

Idea: Find function $u(s, a) \geq d_{\mathcal{F}}(\hat{T}(s, a), T(s, a)$ and define $\tilde{r}(s, a) = r(s, a) - u(s, a)$

Making it work in practice

Uncertainty in ensemble NN

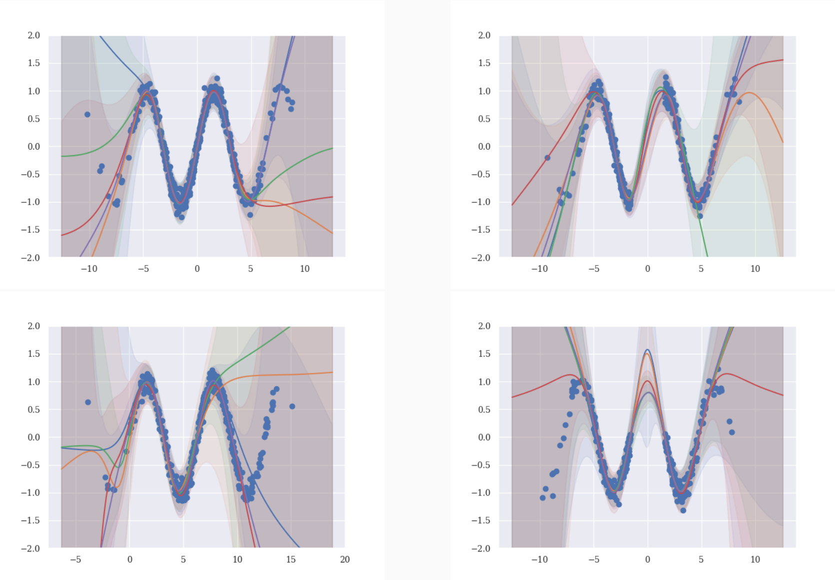
How do we get u?

Core idea (reuses model from PETS, MBPO):

- Take n identical neural networks
- Encode $p(y|x) = \mathcal{N}(\mu^i(s), \Sigma^i(s))$
- Train independently to minimize $-\frac{1}{n}\sum \log p(y|x) | x, y \sim D$
- Each network captures intrinsic randomness (aleatoric)
- Whole ensemble captures data uncertainty (epistemic)

Measure uncertainty with these

Uncertainty in ensemble NN – Visualization



Code available at `https://colab.research.google.com/drive/`

Implementation of a practical algorithm

Require: λ, rollout horizon h, rollout batch size b.

Train on batch data $\mathcal{D}_{\text{env}}$ an ensemble of N probabilistic dynamics $\{\hat{T}^i(s', r|s, a) = \mathcal{N}(\mu^i(s, a), \Sigma^i(s, a))\}_{i=1}^N$.
Initialize policy π and empty replay buffer $\mathcal{D}_{\text{model}} \leftarrow \varnothing$.
for epoch $1, 2, \ldots$ **do**
    for $1, 2, \ldots, b$ (in parallel) **do**
        Sample state s_1 from $\mathcal{D}_{\text{env}}$ for init
        for $j = 1, 2, \ldots, h$ **do**
            Sample an action $a_j \sim \pi(s_j)$.
            Pick $\hat{T}$ from $\{\hat{T}^i\}_{i=1}^N$ and sample $s_{j+1}, r_j \sim \hat{T}(s_j, a_j)$.
            Compute $\tilde{r}_j = r_j - \lambda \max_{i=1}^N \|\Sigma^i(s_j, a_j)\|_F$.
            Add sample $(s_j, a_j, \tilde{r}_j, s_{j+1})$ to $\mathcal{D}_{\text{model}}$
    Drawing samples from $\mathcal{D}_{\text{env}} \cup \mathcal{D}_{\text{model}}$, update π.

Does this work with the theory

We have:

$$N \text{ probabilistic dynamics } \{\hat{T}^i(s', r|s, a) = \mathcal{N}(\mu^i(s, a), \Sigma^i(s, a))\}_{i=1}^N$$

We estimate $\tilde{r}$ as

$$\tilde{r}_j = r_j - \lambda \max_i \|\Sigma^i(s_j, a_j)\|_\mathrm{F} = r(s, a) - \gamma u(s, a)$$

Reminder:

$$|G^\pi_{\hat{T}}(s, a)| \leq d_\mathcal{F}(\hat{T}(s, a), T(s, a)) \stackrel{?}{=} \lambda \max_i \|\Sigma^i(s_j, a_j)\|_\mathrm{F}$$

Fixing bits and pieces

Uncertainty estimate proposed in paper and tested:

$$u(s,a) = \lambda \max_{i=1} \|\Sigma^i(s_j, a_j)\|_F$$

$$u(s,a) = \lambda \max_{i,j} ||\mu_i - \mu_j||_2$$

In experiments, max variance performed better then disagreement...

What about (alternative proposal):

$$u(s,a) = \lambda \mathrm{Var}(\text{ensemble})(s,a) = \lambda \left(\sum \sigma_i^2(s,a) + \sum \mu_i^2(s,a) - \left(\sum \mu_i(s,a) \right)^2 \right)$$

Open question: Relationship of uncertainty and divergence measure

What do we take away?

Summary

- Interesting theory, solid foundation
    - Model-based RL can shine in offline settings
    - Clear connection between model error and expected return
- Empirically very strong algorithm
    - Works very well when requiring OOD data for optimal policy
    - Results mostly skipped here because there were no nice graphs
- Very little connection between theory and empirical work (also noted by reviewers)
- Uncertainty measurement drives even larger gaps between theory and empirical algorithm