Weighted model estimation for offline model-based reinforcement learning

Toru Hishinuma and Kei Senda
Kyoto University

Background

- Offline model-based reinforcement learning.
  (RL using environment model and previously collected offline data)

- Typical approach:
  1. Estimating model by standard supervised learning.
  2. Planning policy using estimated model.

- Issue:
  Model estimation without considering covariate shift.
  * Training data (offline data) is sampled using data-collecting policy.
  * Test data (real future data) is sampled using newly planned policy.

Key idea

- Importance-weighted model estimation can improve predictive performance under covariate shift.

$$-\sum w(s,a)\ \ln P_\theta(s'|s,a)$$

- Natural idea:

$$w(s,a) = \frac{\text{distribution of real future data}}{\text{distribution of offline data}}$$

- Our idea:

$$w(s,a) = \frac{\text{distribution of simulated future data}}{\text{distribution of offline data}}$$

Key idea

- Importance-weighted model estimation can improve predictive performance under covariate shift.

$$-\sum w(s,a)\ \ln P_\theta(s'|s,a)$$

- Natural idea:

$$w(s,a) = \frac{\text{distribution of real future data}}{\text{distribution of offline data}}$$

We cannot access real future data. Estimating this weight is one of challenges in off-policy evaluation in RL. This weight is not easy-to-use.

- Our idea:

$$w(s,a) = \frac{\text{distribution of simulated future data}}{\text{distribution of offline data}}$$

Key idea

- Importance-weighted model estimation can improve predictive performance under covariate shift.

$$-\sum w(s,a)\ \ln P_\theta(s'|s,a)$$

- Natural idea:

$$w(s,a) = \frac{\text{distribution of real future data}}{\text{distribution of offline data}}$$

- Our idea:

$$w(s,a) = \frac{\text{distribution of simulated future data}}{\text{distribution of offline data}}$$

We can generate simulated future data in simulation. Estimating this weight is ratio estimation in supervised learning. This weight is easier-to-use.

Question

Our idea may not seem natural from viewpoint of covariate shift. Is it valid?

Justification

Our idea can be seen evaluating upper bound of policy evaluation error.

$$|\eta(P_\theta, \pi) - \eta(P_*, \pi)| \leq B\sqrt{E[-w(s,a) \ln P_\theta(s'|s,a)] - \text{const}}$$

Justification: derivation

Policy evaluation error

$$
\begin{aligned}
\left|\eta^{\pi}_{\mathcal{P}_*} - \eta^{\pi}_{\mathcal{P}_\theta}\right| &\leq \frac{\gamma \mathbb{E}_{(s,a)\sim d^{\pi}_{\mathcal{P}_\theta}}\left[\left|\sum_{s'}\left(\mathcal{P}_*(s'|s,a) - \mathcal{P}_\theta(s'|s,a)\right) V^{\pi}_{\mathcal{P}_*}(s')\right|\right]}{1-\gamma} \\
&\leq \frac{B \mathbb{E}_{(s,a)\sim d^{\pi}_{\mathcal{P}_\theta}}\left[||\mathcal{P}_*(\cdot|s,a) - \mathcal{P}_\theta(\cdot|s,a)||_1\right]}{\sqrt{2}} \\
&\leq B\sqrt{\mathbb{E}_{(s,a)\sim d^{\pi}_{\mathcal{P}_\theta}}\left[c_\theta(s,a) - h(s,a)\right]} \\
&= B\sqrt{\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}_{\mathcal{P}_*}}\left[w^{\pi}_\theta(s,a)c_\theta(s,a)\right] - \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}_{\mathcal{P}_*}}\left[w^{\pi}_\theta(s,a)h(s,a)\right]} \\
&\leq B\sqrt{\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}_{\mathcal{P}_*}}\left[w^{\pi}_\theta(s,a)c_\theta(s,a)\right] - h_{\min}},
\end{aligned}
$$

Telescoping Lemma [Luo+2019]

Holder inequality

Pinsker inequality + Jensen inequality

Definition of minimum self-entropy

$$
\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}_{\mathcal{P}_*}}\left[w^{\pi}_\theta(s,a)c_\theta(s,a)\right] = \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}_{\mathcal{P}_*}, s'\sim\mathcal{P}_*(\cdot|s,a)}\left[-w^{\pi}_\theta(s,a)\ln\mathcal{P}_\theta(s'|s,a)\right]
$$

$$
\fallingdotseq -\frac{1}{N}\sum_{n=1}^{N} w^{\pi}_\theta(s_n,a_n)\ln\mathcal{P}_\theta(s'_n|s_n,a_n)
$$

Importance-weighted loss function

Loss functions

- (Repeat) upper bound of policy evaluation error

$$|\eta(P_\theta, \pi) - \eta(P_*, \pi)| \leq B\sqrt{E[-w(s,a)\ln P_\theta(s'|s,a)] - \text{const}}$$

- Policy evaluation:

$$L(\theta) = E[-w(s,a)\ln P_\theta(s'|s,a)]$$

- Policy optimization:

$$J(\theta, \pi) = \eta(P_\theta, \pi) - B'\sqrt{E[-w(s,a)\ln P_\theta(s'|s,a)] - \text{const}}$$

Algorithm: policy evaluation (full version)

- Loss function:

$$L(\theta) = E\left[-w_\theta^\pi(s,a)\ln P_\theta(s'|s,a)\right] \approx -\sum w_\theta^\pi(s,a)\ln P_\theta(s'|s,a)$$

- Gradient-based optimization:

$$\nabla L(\theta) \approx -\sum w_\theta^\pi(s,a)\{\nabla \ln P_\theta(s'|s,a) + \ln P_\theta(s'|s,a)\,\nabla \ln d(s)\}$$

Ratio estimation for supervised learning

Extension of LSDG [Morimura+2010]

Algorithm: policy evaluation (simplified version)

- (Repeat) gradient:

$$\nabla L(\theta) \approx -\sum w_\theta^\pi(s,a)\{\nabla \ln P_\theta(s'|s,a) + \ln P_\theta(s'|s,a)\,\nabla \ln d(s)\}$$

Estimating $\nabla \ln d(s)$ is computationally heavy, because it is the same number of value function of forward Bellman equation as the number of parameters.

- Simplified version:

$$-\sum w_\theta^\pi(s,a)\nabla \ln P_\theta(s'|s,a)$$

Algorithm: policy optimization

- Loss function:

$$J(\theta, \pi) = \eta(P_\theta, \pi) - B'\sqrt{E[-w(s,a) \ln P_\theta(s'|s,a)] - \text{const}}$$

Model expected return

Penalty for policy evaluation error

- EM-style optimization of majorization-minimization surrogate of $J(\theta, \pi)$:

  E-step: modification of weighted model estimation for policy evaluation.

  M-step: policy optimization in simulated MDP with penalized reward.

Pendulum swing-up prediction using small NNs

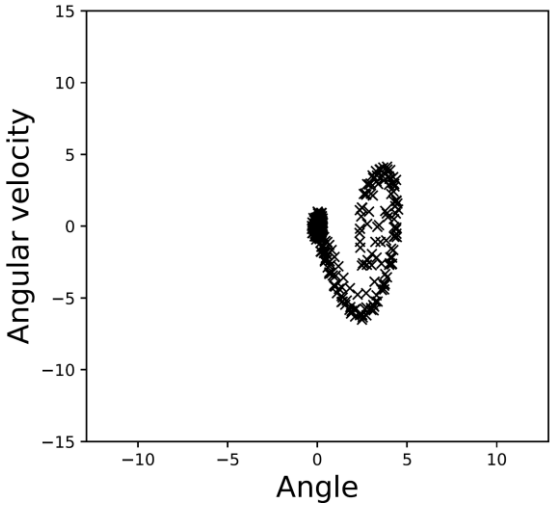

Fig (a)

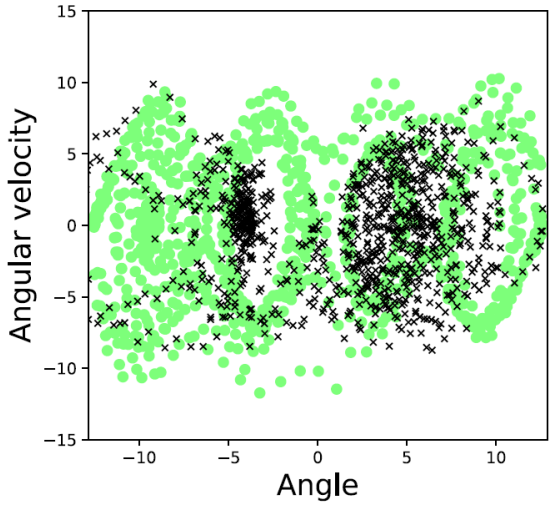

Fig (b)

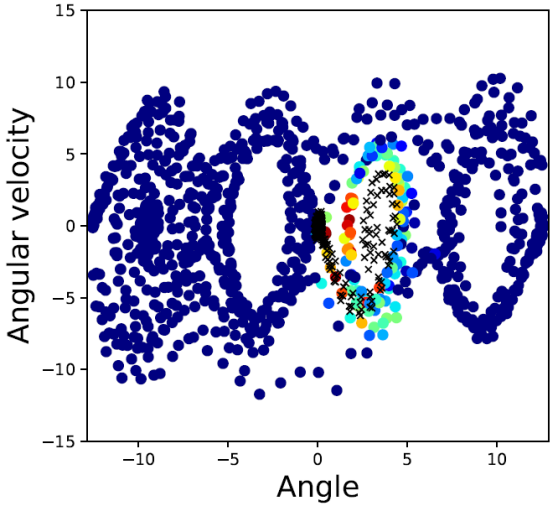

Fig (c)

Fig (a) shows real future data obtained using optimal policy.

The goal is to predict swing-up.

Fig (b) shows result of standard supervised learning.
Fig (c) shows result of weighted model estimation.

Black markers are simulated future data.
Colored markers are offline data, where coloring indicates importance weighting.

× Fig (b) cannot capture swing-up, because small NNs cannot generalize globally.
○ Fig (c) can capture swing-up, because small NNs can generalize locally around swing-up behavior, based on importance weighting.

D4RL MuJoCo benchmark

Our EM-style algorithm

dataset	CQL [37]	original MOPO [8]	$\alpha = 0$	$\alpha = 0.2$
HalfCheetah-random	35.4	35.4 ± 2.5	48.7 ± 2.8	49.1 ± 3.2
HalfCheetah-medium	44.4	42.3 ± 1.6	75.7 ± 1.5	73.1 ± 5.2
HalfCheetah-medium-replay	46.2	53.1 ± 2.0	72.1 ± 1.4	65.5 ± 6.4
HalfCheetah-medium-expert	62.4	63.3 ± 38.0	73.9 ± 24.2	85.7 ± 21.6
Hopper-random	10.8	11.7 ± 0.4	30.2 ± 4.4	32.7 ± 0.5
Hopper-medium	86.6	28.0 ± 12.4	100.9 ± 2.7	104.1 ± 1.2
Hopper-medium-replay	48.6	67.5 ± 24.7	97.2 ± 10.9	104.0 ± 3.2
Hopper-medium-expert	111.0	23.7 ± 6.0	109.3 ± 1.1	104.9 ± 10.1
Walker2d-random	7.0	13.6 ± 2.6	16.5 ± 6.6	18.4 ± 7.6
Walker2d-medium	74.5	17.8 ± 19.3	81.7 ± 1.2	60.7 ± 29.0
Walker2d-medium-replay	32.6	39.0 ± 9.6	80.7 ± 3.1	82.7 ± 3.3
Walker2d-medium-expert	98.7	44.6 ± 12.9	59.5 ± 49.4	108.2 ± 0.5

- Our EM-style algorithm improves performance for walker2d-medium-expert dataset.

Limitation

- Full version of our algorithm is derived from theory, but it cannot be applied to large-scale problems due to large amount of computation.

- Simplified version of our algorithm is practical, but it has no convergence with respect to loss function.

Conclusion

- We discuss model estimation considering covariate shift in offline MBRL.
- Our idea is importance-weighting with distribution ratio of offline data and simulated future data.
- Question: our idea may not seem natural. Is it valid?
    - Our idea is justified as evaluating upper bound of policy evaluation error.
    - We propose EM-style algorithm based on our idea. It improves performance in numerical experiments.

Future issues

- Extension to Bayesian MBRL.
- Combining with loss functions in decision-aware model learning approaches.
- Model selection based on importance-weighted loss function.
- Addressing extrapolation.

References

- Levine et al. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. 2020.
- Luo et al. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. 2019.
- Morimura et al. Derivatives of logarithmic stationary distributions for policy gradient reinforcement learning. 2010.
- Yu et al. MOPO: Model-based offline policy optimization. 2020.