Weighted model estimation for offline model-based reinforcement learning

Toru Hishinuma and Kei Senda
Kyoto University

Background

- オフライン・モデルベース・強化学習
  - 強化学習　　　…　データから方策／制御器を学習する
  - オフライン　　…　あらかじめ収集しておいたオフラインデータのみを使う
  - モデルベース　…　環境モデルを陽に推定して利用する

- よくある手順：
  1. 経験損失最小化（例：最小二乗法）でモデルを推定する
  2. 推定したモデルを使って方策をプランニングする

- 課題：

  共変量シフトによりモデルの予測性能が低下　→　方策のプランニングに影響
  - 訓練データ　　…　オフラインデータ収集方策に従ってサンプルされる
  - テストデータ　…　エージェントの将来の方策に従ってサンプルされる

Approach

- 一般に、重み付け経験損失最小化で、共変量シフト下での予測性能を改善できる。
  オフライン・モデルベース・強化学習でも、同じようなことをやりたい。

$$L(\theta) = -\sum w(x_n) \ln P_\theta(y_n | x_n)$$

- 自然な発想：

$$w(x) = \frac{\text{将来の方策に対応する実環境データの分布}}{\text{オフラインデータの分布}}$$

将来実環境データは直接使えず、その推定自体がオフ方策評価の主要課題。なかなか難しい。

- 本研究：

$$w(x) = \frac{\text{将来の方策に対応するシミュレーションデータの分布}}{\text{オフラインデータの分布}}$$

シミュレーションデータを生成し、それを使って密度比推定すれば求まる。比較的簡単。

Research question

「シミュレーションデータで置き換えた重み付けでも大丈夫か？」

- 再掲：
  - 自然な発想 … $w(x) = \dfrac{\text{将来の方策に対応する実環境データの分布}}{\text{オフラインデータの分布}}$
  - 本研究 … $w(x) = \dfrac{\text{将来の方策に対応するシミュレーションデータの分布}}{\text{オフラインデータの分布}}$

Justification (1/2)

本研究の重み付け損失関数は、方策評価誤差のupper boundを評価している

$$\underbrace{|\eta(P_\theta,\pi)}_{\text{期待リターン（モデル上）}} - \underbrace{\eta(P_*,\pi)|}_{\text{期待リターン（実環境上）}} \leq B\sqrt{\underbrace{E\left[-w_\theta^\pi(s,a)\ln P_\theta(s'|s,a)\right]}_{\text{本研究の重み付け損失関数}} - \mathrm{const.}}$$

→　重み付け損失を減らせば方策評価誤差も減るので、理屈的には良さそう

Justification (2/2)

方策評価誤差

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

Telescoping Lemma [Luo+2019]

Holder不等式+価値関数上限

Pinsker不等式+Jensen不等式

遷移確率自己エントロピーの最小値（定数）

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

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

本研究の重み付け経験損失

Loss function

- 方策評価の損失関数（モデルパラメータθのみを最適化）：

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

- 方策最適化の損失関数（モデルパラメータθと方策πを最適化）：

$$J(\theta,\pi) = \underbrace{-\eta(P_\theta,\pi)}_{\text{期待リターン（モデル上）}} + B'\underbrace{\sqrt{E\left[-w_\theta^\pi(s,a)\ln P_\theta(s'|s,a)\right] - \mathrm{const}}}_{\text{方策評価誤差に対するペナルティ}}$$

Algorithm: weighted model estimation for policy evaluation (1/2)

- 方策評価の損失関数

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

- 勾配

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

密度比推定
分子：シミュレーションデータ
分母：オフラインデータ

LSDG[Morimura+2010]の拡張

Algorithm: weighted model estimation for policy evaluation (2/2)

- 再掲：勾配

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

- アルゴリズム：
  - モデルパラメータ初期化：　重み無し経験損失最小化などで、θを与える
  - 収束するまで以下を繰り返す
    - シミュレーション：　モデルパラメータθのモデルで、シミュレーションデータを生成
    - 密度比推定：　シミュレーションデータ＋オフラインデータで、w_θ^πを推定
    - LSDGの拡張：　モデルパラメータθのモデル上で、 $\nabla \ln d_\theta^\pi$を推定
    - モデルパラメータ更新：　損失関数と勾配に基づいて、θを更新

Pendulum swing-up prediction using small NNs

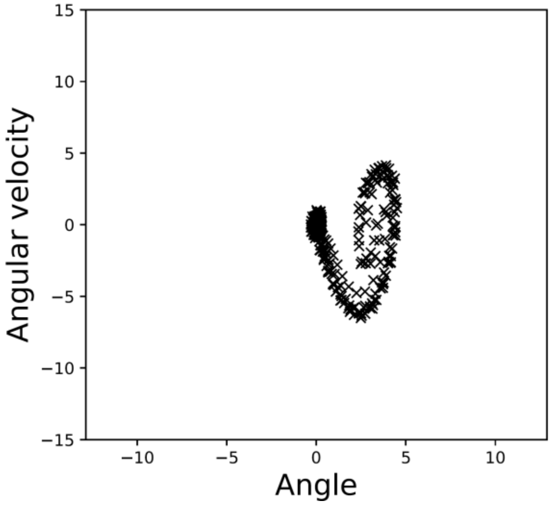


Fig (a)

- 将来の実データ
- 横軸：角度、縦軸：角速度
- 原点に向かって振り上げる挙動であり、これをモデルで予測したい

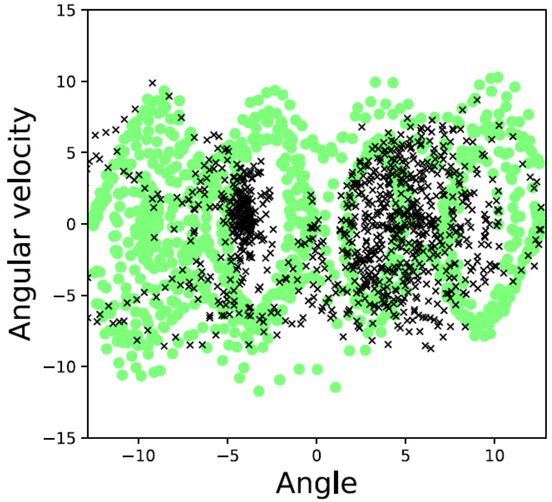


Fig (b)

- 黒：重み無し推定モデルの将来予測
- カラー：訓練データ重み
  → 重みは一様
- 振り上げ挙動が予測できていない
  ↑ 表現能力不足で汎化できない

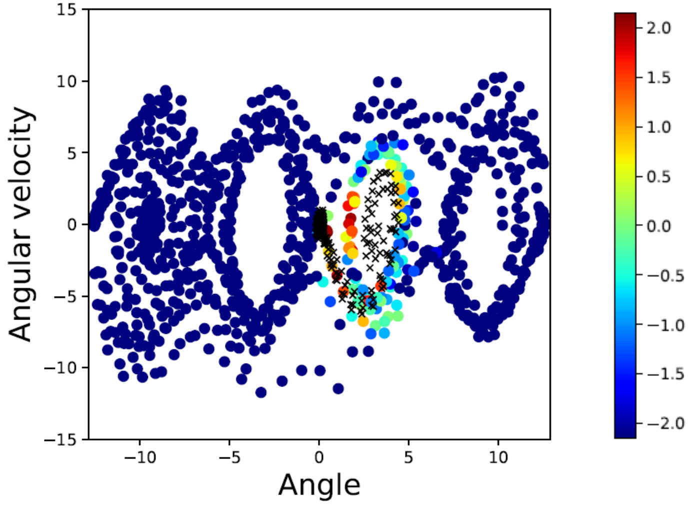
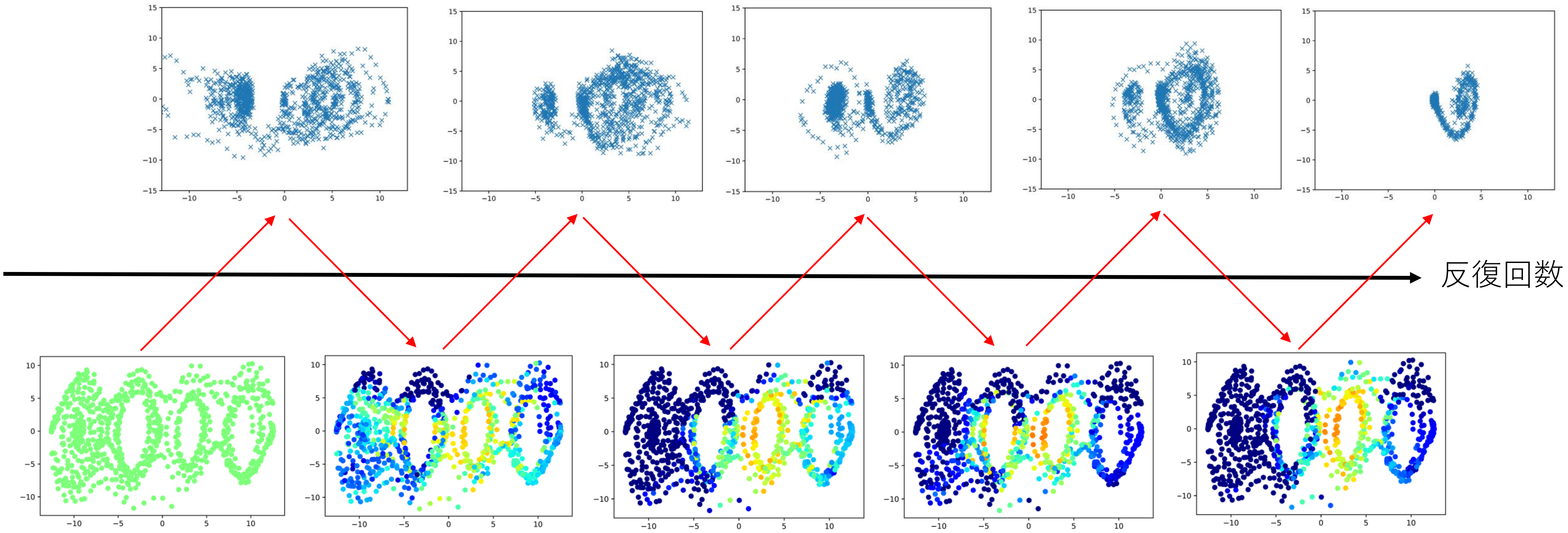


Fig ©

- 黒：重み付け推定モデルの将来予測
- カラー：訓練データ重み
  → 振り上げ挙動の周りに大きい重み
- 振り上げ挙動が予測できている
  ↑ 振り上げ挙動周りを精度よく予測

Pendulum swing-up prediction using small NNs



Algorithm: weighted model estimation for policy evaluation (simplified version)

- 再掲：勾配

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

$\nabla \ln d_\theta^\pi$の推定にはかなり計算量が必要で、大規模問題だと辛い。
具体的には、モデルパラメータと同数のMDPに対する価値関数推定が必要。

- 簡略化アルゴリズム：
  - モデルパラメータ初期化：
  - ~~収束するまで以下を繰り返す~~
  - 収束しないので、損失改善具合を見つつ以下を繰り返す（本研究の欠点…）
    - シミュレーション：　モデルパラメータθのモデルで、シミュレーションデータを生成
    - 密度比推定：　シミュレーションデータ＋オフラインデータで、w_θ^πを推定
    - ~~LSDGの拡張：　モデルパラメータθのモデル上で、$\nabla \ln d_\theta^\pi$を推定~~
    - モデルパラメータ更新：　損失関数と勾配に基づいて、θを更新

Algorithm: policy optimization based on weighted model estimation

- 再掲：方策最適化の損失関数（モデルパラメータθと方策πを最適化）

$$J(\theta,\pi) = \underbrace{-\eta(P_\theta,\pi)}_{\text{期待リターン（モデル上）}} + \underbrace{B'\sqrt{E\left[-w_\theta^\pi(s,a)\ln P_\theta(s'|s,a)\right] - \mathrm{const}}}_{\text{方策評価誤差に対するペナルティ}}$$

- EM法で、 $J(\theta,\pi)$ を（代理関数を通じて）最大化する
  - E ステップ：前述の重み付けモデル推定法（の微修正）で、モデルパラメータ θ を更新
  - Mステップ：モデルとペナルティ付き報酬によるシミュレーションMDP上で、方策 π を更新

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

- walker2d-medium-expert datasetに対しては、性能を改善した
- （それ以外に対しては、特に性能改善しなかった）

Conclusion

- オフライン・モデルベース・強化学習で、共変量シフトを考慮した重要度重み付けモデル推定の方法を議論した。

$$\text{本研究の重み付け} = \frac{\text{将来の方策に対応する}\cancel{\text{実データ}}\text{シミュレーションデータの分布}}{\text{オフラインデータの分布}}$$

- 研究課題：シミュレーションデータで置き換えた重み付けでも大丈夫か？
  - 理屈的にはよさそう　←　重み付け損失が方策評価誤差のupper boundになっている
  - 実際的にもよさそう　←　アルゴリズムを作って適用して、数値実験で改善が見られた

Future issues

- ベイズ・モデルベース・強化学習への拡張

  ↑元々これをやりたくて、本研究はその途中経過（重み付き尤度の定義までやれた）という感じ

- 大規模タスクに対する$\nabla \ln d_\theta^\pi$の推定（今回無視したやつ）

  ↑報酬関数が異なるMDPが多数ある場合に、それぞれの前進Bellman方程式の解を推定したい

  ↑マルチタスク強化学習みたいなやつを使う？

- オフラインデータのカバー範囲と密度比推定に関する議論

  ↑今回、このことを全然ケアしていなかった（ポスター会場で指摘されて気づいた…）

- 他の外挿手法との組み合わせ

  ↑D4RLだと結果が期待したほど改善しなかったので…

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.

Ratio estimation

- 確率的分類器による密度比推定法

$$w(x) = \frac{p_{\text{test}}(x)}{p_{\text{train}}(x)} = \frac{p(y = \text{"train"})}{p(y = \text{"test"})} \frac{p\ (y = \text{"test"}|x)}{p\ (y = \text{"train"}|x)} \approx \frac{n_{\text{train}}}{n_{\text{test}}} \frac{\hat{p}\ (y = \text{"test"}|x)}{\hat{p}(y = \text{"train"}|x)}$$

- 重要度重み付き経験損失最小化の安定化

$$w'(x) = [w(x)]^{\alpha}$$

  - $w(x)$をそのまま使うIWERMは、一致性を持つが不安定になり得る [Shimodaira2000]
  - $\alpha <$ １である$w'(x)$を使えば、（一致性を犠牲にして）安定化することができる

Surrogate function for EM-style optimization

方策最適化の損失関数

$$J(\theta,\pi) = -\eta(P_\theta,\pi) + B' \sqrt{E\left[-w_\theta^\pi(s,a)\ln P_\theta(s'|s,a)\right] - \mathrm{const}}$$

上界最小化手法をルートの部分に適用すると、

$$J_{\mathrm{surr}}(\theta,\pi) = -\eta(P_\theta,\pi) + b'E\left[-w_\theta^\pi(s,a)\ln P_\theta(s'|s,a)\right] - \mathrm{const}$$

$$= E_{(s,a)\sim d_{\mathrm{offline\_data}}}\left\{w_\theta^\pi(s,a)\left[\frac{r(s,a)}{1-\gamma} - b'\ln P_\theta(s'|s,a)\right]\right\} - \mathrm{const}$$

Derivative of log-stationary distribution

割引MDPにおける定常状態分布は、

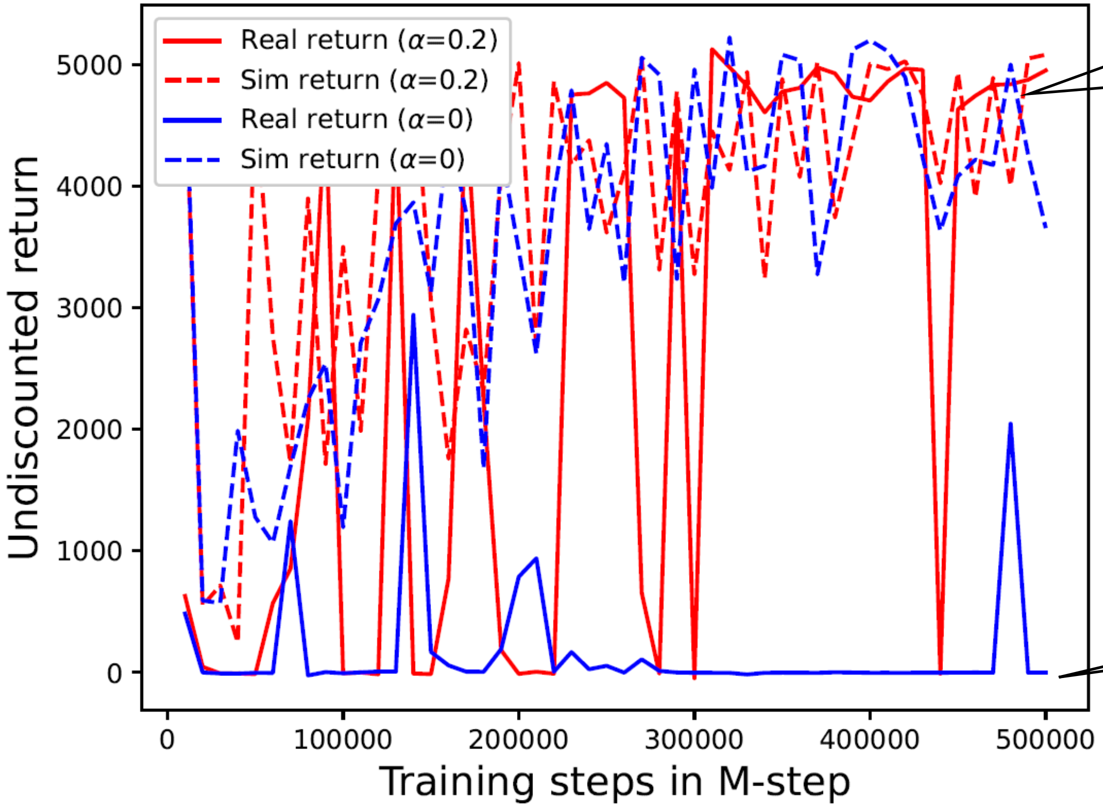
$$d_\theta^\pi(s') = (1-\gamma)\rho(s') + \gamma \sum_{s,a} d_\theta^\pi(s)\pi(a|s)P_\theta\,(s'|s,a)$$

i番目のパラメータθ_iで微分すると、前進Bellman方程式が得られる

$$d_\theta^\pi(s')\underbrace{\nabla_{\theta_i}\ln d_\theta^\pi(s')}_{\text{価値関数}} = \sum_{s,a} d_\theta^\pi(s)\pi(a|s)P_\theta\,(s'|s,a)\left\{\gamma\underbrace{\nabla_{\theta_i}\ln P_\theta\,(s'|s,a)}_{\text{報酬}} + \gamma\underbrace{\nabla_{\theta_i}\ln d_\theta^\pi(s)}_{\text{価値関数}}\right\}$$

D4RL MuJoCo benchmark

- walker2d-medium-expert dataset



赤色（重み付けモデル推定をする方）は、モデルで推定した期待リターンと実環境の期待リターンが比較的近い。

青色（重み無しモデル推定をする方）は、モデルで推定した期待リターンと実環境の期待リターンとの差が比較的大きい。具体的には、「モデル上だと最後まで乾燥するが、実環境だと一歩目でコケる」みたいな方策が学習されている。