When to Trust Your Model: Model-Based Policy Optimization

22/12/12

김도윤

논문 정보

- 제목: When to Trust Your Model: Model-Based Policy Optimization
- 저자: Michael Janner, Justin Fu, Marvin Zhang, Sergey Levine (UC Berkeley)
- 출판: NIPS2019

When to trust your model: Model-based policy optimization

M Janner, J Fu, M Zhang… - Advances in Neural …, 2019 - proceedings.neurips.cc

Designing effective model-based reinforcement learning algorithms is difficult because the ease of data generation must be weighed against the bias of model-generated data. In this paper, we study the role of model usage in policy optimization both theoretically and empirically. We first formulate and analyze a model-based reinforcement learning algorithm with a guarantee of monotonic improvement at each step. In practice, this analysis is overly pessimistic and suggests that real off-policy data is always preferable to model-generated …

논문 요약

- 기존 Model-based RL의 문제였던 rollout이 길 때의 문제를 이론적, 실험적으로 향상
- MoJuCo Benchmark를 통해 다른 Model-Free 및 Model-based와 비교

Model-Free Vs Model-based

- Model-Free의 경우 generalization이 쉬우나 sample efficiency가 아쉽다
- 그래서 도입된 것이 Model-based
    - Gaussian process, Time-varying linear dynamical systems → Neural network
    - env의 true dynamics를 따라하는 model을 만들어 data를 만들어 낸다. (rollout)
    - real experience와 generated experience를 쓰기 때문에 sample efficiency가 좋다.
    - 그러나 model error가 크다면 policy improvement를 보장할 수 없다.
        - model usage의 비율을 조절해야 한다.

Background (Terminology)

- optimal policy는 다음과 같다.

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \eta[\pi] = \underset{\pi}{\operatorname{argmax}}\; E_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right]$$

- True dynamics $p(s'|s,a)$ → model $p_\theta(s'|s,a)$
  - supervised learning으로 배운다.

Monotonic Improvement with model bias

Algorithm 1 Monotonic Model-Based Policy Optimization

1: Initialize policy $\pi(a|s)$, predictive model $p_\theta(s', r|s, a)$, empty dataset $\mathcal{D}$.
2: **for** N epochs **do**
3: Collect data with π in real environment: $\mathcal{D} = \mathcal{D} \cup \{(s_i, a_i, s'_i, r_i)\}_i$
4: Train model p_θ on dataset $\mathcal{D}$ via maximum likelihood: $\theta \leftarrow \text{argmax}_\theta \mathbb{E}_{\mathcal{D}}[\log p_\theta(s', r|s, a)]$
5: Optimize policy under predictive model: $\pi \leftarrow \text{argmax}_{\pi'} \hat{\eta}[\pi'] - C(\epsilon_m, \epsilon_\pi)$

- Optimize policy under learned model ↔ Collect data under the updated policy
- 이 때 error가 유발되서 policy quality를 떨어뜨리는데
  - Error(Bias)가 있는 Model을 사용(exploitation)해서 policy를 optimization할 때
  - Predicted return과 True return이 너무도 다를 때

Monotonic Improvement with model bias

$$\eta[\pi] \geq \hat{\eta}[\pi] - C.$$

- 위와 같이 true return과 model return의 차이의 bound를 C라고 할 때,
- model return을 C보다 크게 향상 시키면 true return도 향상될 것이다!
- 그렇다면 일단 C를 어떻게 수식으로 표현할 수 있을까?
-

Theorem 4.1

$$\eta[\pi] \geq \hat{\eta}[\pi] - \underbrace{\left[\frac{2\gamma r_{\max}(\epsilon_m + 2\epsilon_\pi)}{(1-\gamma)^2} + \frac{4 r_{\max}\epsilon_\pi}{(1-\gamma)}\right]}_{C(\epsilon_m, \epsilon_\pi)}$$

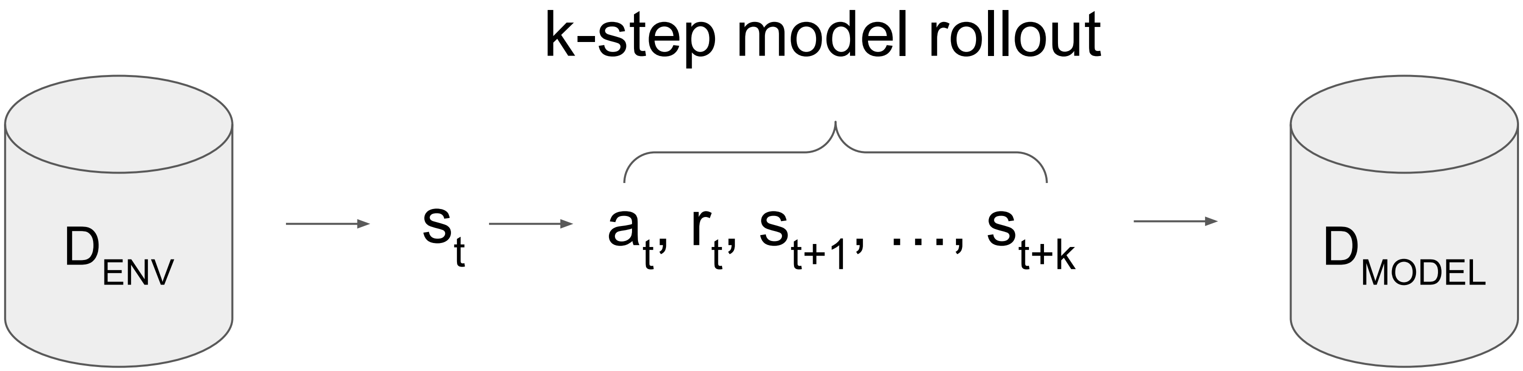
- Model Error (ϵ_m) $\epsilon_m = \max_t E_{s \sim \pi_{D,t}}[D_{TV}(p(s', r|s, a)||p_\theta(s', r|s, a))]$
    - Generalization error due to sampling: 모든 경우의 수(혹은 state distribution)을 가지고 model을 만든 것이 아니기 때문에 model은 approximation error를 가질 수 밖에 없다.
    - Supervised learning으로 배우는 것이므로 PAC generalization bound라는 것으로 표현 가능
- Policy Error (ϵ_π) $\max_s D_{TV}(\pi||\pi_D) \leq \epsilon_\pi$
    - distribution shift due to updated policy: model을 만들 때 썼던 policy가 만들어내는 state distribution과 현재 policy가 만들어내는 state distribution은 당연히 다르다.
    - Pinsker's inequality로 표현 가능

Monotonic Improvement with model bias

$$\eta[\pi] \geq \hat{\eta}[\pi] - \underbrace{\left[\frac{2\gamma r_{\max}(\epsilon_m + 2\epsilon_\pi)}{(1-\gamma)^2} + \frac{4 r_{\max}\epsilon_\pi}{(1-\gamma)}\right]}_{C(\epsilon_m, \epsilon_\pi)}$$

- 위 부등식의 문제점
  - 여기서 문제는 ϵ_m이 굉장히 크다면 위 부등식을 만족하는 pi를 찾을 수 없어 improvement를 보장할 수 없다.
  - rollout이 full일 때를 가정한 것이다.

Branched Rollout



true dynamics에서 나온 data.
model을 학습할 때 쓰인다.

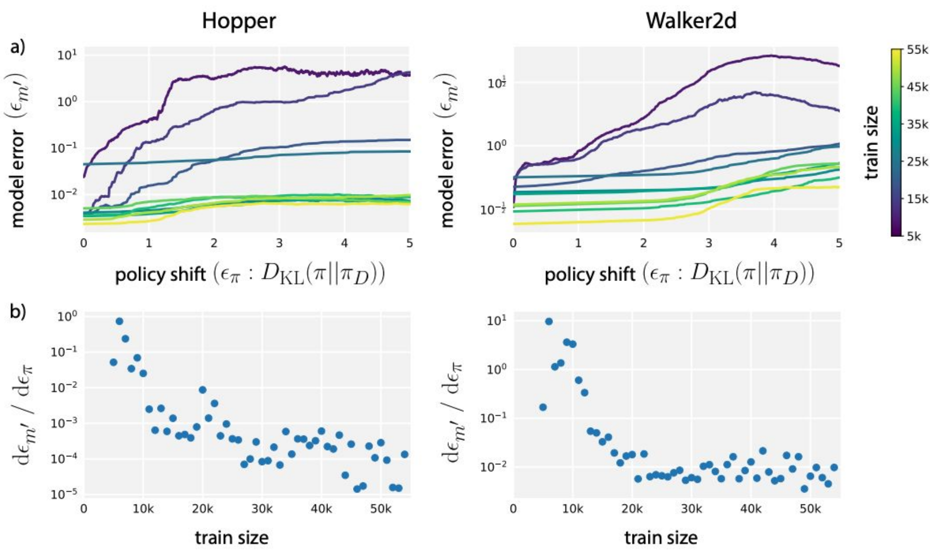
model이 만든 데이터
policy optimization에 쓰인다.

Theorem 4.2

$$\eta[\pi] \geq \eta^{\text{branch}}[\pi] - 2r_{\max}\left[\underbrace{\frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^k + 2}{(1-\gamma)}}_{\text{policy error}}\epsilon_\pi + \underbrace{\frac{k}{1-\gamma}(\epsilon_m + 2\epsilon_\pi)}_{\text{model error}}\right]$$

- policy error: k가 커지면 off-policy error가 줄어든다. (시작이 off-policy인데 current policy로한 경험들이 model에 의해 추가된다.)
- model error: k가 커지면 커진다.
- Theorem 4.2로 optimal k를 구할 수 있다.
- 그러나 문제는 ϵ_m을 ϵ_π에 비해 pessimistic하게 scaling
- Empirically measure해서 어떤지 알아보자.

Model Generalization in practice



- ϵ_π가 변할 때 ϵ_m는 어떻게 변할까?
- 즉, 우리가 만든 model이 model을 만들 때 썼던 policy 외에서도 잘 적용이 될까?
- model을 만들 때 썼던 data가 많을수록 model error가 잘 커지지 않는다.

Model Generalization in practice

$$\hat{\epsilon}_{m'}(\epsilon_\pi) \approx \epsilon_m + \epsilon_\pi \frac{\mathrm{d}\epsilon_{m'}}{\mathrm{d}\epsilon_\pi}$$

- 위 실험을 참고해서 복잡하게 수식으로 표현된 model error(pessimistic)을 linear model로 표현해 보자.
- 이전에는 model error를 가지고 current policy distribution에 대한 error를 $\epsilon_m + 2\epsilon_\pi$로 근사

Theorem 4.3

$$\eta[\pi] \geq \eta^{\text{branch}}[\pi] - 2r_{\max}\left[\frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^k\epsilon_\pi}{(1-\gamma)} + \frac{k}{1-\gamma}(\epsilon_{m'})\right]$$

$$k^* = \underset{k}{\operatorname{argmin}}\left[\frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^k\epsilon_\pi}{(1-\gamma)} + \frac{k}{1-\gamma}(\epsilon_{m'})\right] > 0$$

- 충분히 작은 ϵ_m'에서는 위와 같이 optimal k를 구할 수 있다.

Implementation

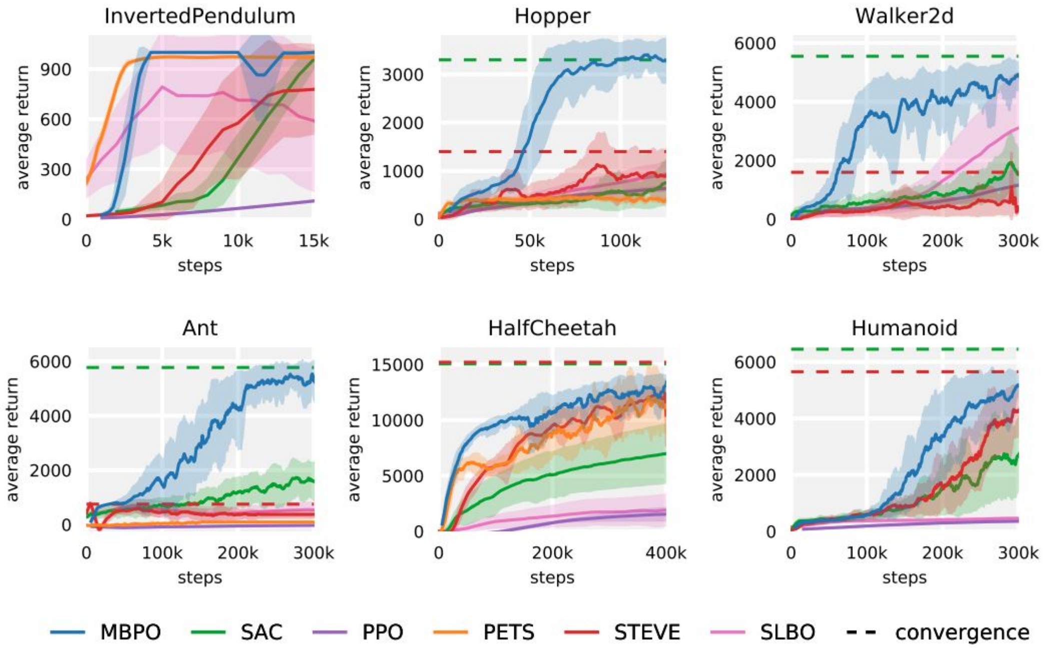
- predictive model: bootstrap ensemble of dynamic models
- policy optimization: SAC
- model usage: branched strategy

Algorithm 2 Model-Based Policy Optimization with Deep Reinforcement Learning

Initialize policy π_ϕ, predictive model p_θ, environment dataset $\mathcal{D}_{\text{env}}$, model dataset $\mathcal{D}_{\text{model}}$
for N epochs **do**
    Train model p_θ on $\mathcal{D}_{\text{env}}$ via maximum likelihood
    for E steps **do**
        Take action in environment according to π_ϕ; add to $\mathcal{D}_{\text{env}}$
        for M model rollouts **do**
            Sample s_t uniformly from $\mathcal{D}_{\text{env}}$
            Perform k-step model rollout starting from s_t using policy π_ϕ; add to $\mathcal{D}_{\text{model}}$
        for G gradient updates **do**
            Update policy parameters on model data: $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi, \mathcal{D}_{\text{model}})$

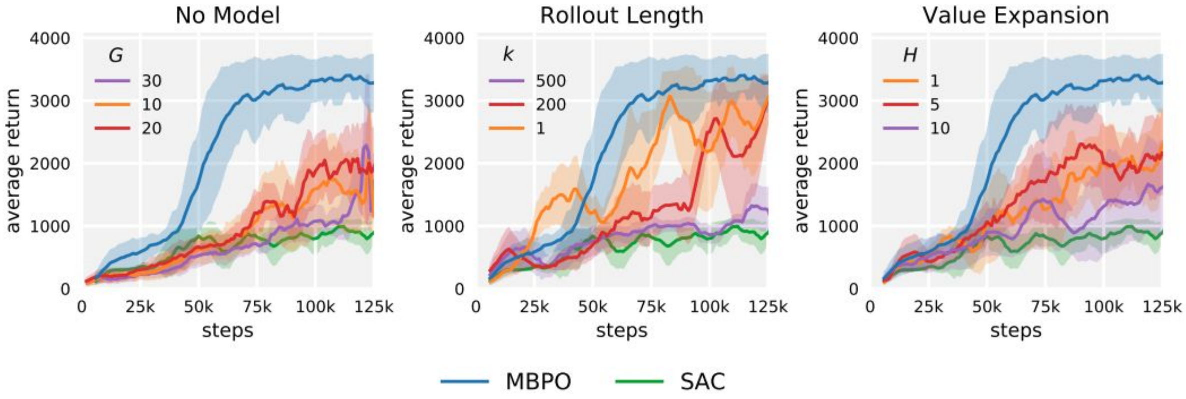
Performance Comparison

*MuJuCo



● 기존 model-free, 기존 model-based를 모두 이긴다.

Design Evaluation



- gradient update 회수를 비슷하게 model-free에서 늘려봤지만 여전히 MBPO가 낫다.
- rollout 길이를 바꿔봤는데 신기하게도 k=1도 꽤나 도움이 되고 k=500정도까지 늘리면 model error가 커서 성능이 안좋다.
- Model-based 중에서도 value expansion type을 SAC와 구현해 봤는데 여전히 MBPO가 좋다.

Model Exploitation problem

- model error가 compound하면 어쩌나 했는데
- rollout 길이가 충분히 짧으면 실제 dynamic이랑 비슷해서 괜찮다.
- 오른쪽 그림은 return이 어떻게 비슷한지를 보여준다.