When to Trust Your Model: Model-Based Policy Optimization

Yeushyi Huang
Department of Electronics and Electrical Engineering
National Yang Ming Chiao Tung University
eric5642.c@nycu.edu.tw

1 Introduction

This paper mainly studies the role of model usage in policy optimization, then they propose a simple procedure, which is called model-based policy optimization (MBPO), that using short model-generated rollouts branched from real data.

1.1 Main research challenge

Nowadays, RL algorithms generally fall into two types: model-based approaches and model-free techniques. Model-based ones will build a predictive model of an environment, then derive a controller from it; while model-free ones learn directly from state to actions. Both have their own pros and cons.

For model-free algorithms, they have shown that having outstanding performance over general-purpose goal due to raw states input, but their generality come at cost of efficiency.

As for model-based algorithms, they are appealing because of their comparatively fast learning when it comes to arduous data collection. Nevertheless, pretrained-model accuracy often acts as the bottleneck to policy quality, leading model-based algorithms perform worse than model-free counterparts.

So this paper is mainly to figure out how to use a predictive model for policy optimization effectively.

1.2 High level technical insights

To achieve pronounced performance and effective training, they disentangle the task and model horizon by querying the model for only sort rollouts.

1.3 Main contribution of this paper

They proposes model-based policy optimization (MBPO). MBPO adapts an ensemble of probabilistic networks, and it's shown that combination of model ensembles with short rollouts is sufficient to prevent model exploitation.

They demonstrate that large amount of short model-generated rollouts can make the algorithm learn substantially fast while avoiding the same pitfalls of prior model-based algorithms including model exploitation and failure on long-horizon task.

2 Problem Formulation

2.1 Notations

- Markov decision process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma, \rho_0)$
  - $\mathcal{S}$: state space
  - $\mathcal{A}$: action space
  - p: $p(s'|s,a)$ denotes the transition distribution
  - r: $r(s,a)$ denotes the reward function
  - γ: discount factor, $\gamma \in (0,1)$
  - ρ_0: $\rho_0(s)$ denotes the initial state distribution
- π^*: the optimal policy
- η: the expected sum of discounted rewards
- $p(s'|s,a)$: the dynamics
- $p_\theta(s'|s,a)$: the model of the transition distribution

2.2 Preliminaries and Technical assumptions

1. The goal of reinforcement learning is to find the optimal policy π^* that maximizes the expected sum of discounted rewards, η:

$$\pi^* = \arg\max_\pi \eta[\pi] = \arg\max_\pi \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$$

2. The dynamics $p(s'|s,a)$ are assumed to be unknown, and model-based method is to construct a model of the transition distribution $p_\theta(s'|s,a)$

3 Theoretical Analysis

3.1 Monotonic improvement with model bias

They first lay out a general recipe for MBPO with monotonic improvement as described in Algorithm 1. MBPO follows in 3 steps:

1. MBPO optimizes a policy under a learned model
2. collects data under the updated policy
3. uses that data to train a new model

Algorithm 1 Monotonic Model-based Policy Optimization

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

3.1.1 Monotonic model-based improvement

To show monotonic improvement for model-based method, they construct a bound of the following form:

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

$\eta[\pi]$ denotes the returns of the policy in the true MDP, whereas $\hat{\eta}[\pi]$ denotes the returns of the policy under the model. As they can improve by at least C under the model, they can ensure the improvement on the true MDP.

The returns gap, C, mainly comes from two error quantities: generalization error (Shalev-Shwartz and Ben-David [2014]) due to sampling and distribution shift due to the updated policy encountering not seen before.

generalization error $\epsilon_m = \max_t E_{s\sim\pi_{D,t}}[D_{TV}(p(s',r|s,a)||p_\theta(s',r|s,a))]$

distribution shift $\max_s D_{TV}(\pi||\pi_D) \leq \epsilon_\pi$

With above error sources, the bound:

Theorem 1 *Let the expected TV-distance between two transition distribution be bounded at each timestep by ϵ_m and the policy divergence be bounded by ϵ_π. Then the true returns and model returns of the policy are bounded as:*

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

3.1.2 Interpolating model-based and model-free updates

They introduce the notion of a *branched rollout* to rely less on the model and instaed more on real data collected from the true dynamics. This branched rollout structure resembles Dyna algorithm (Sutton [1990]), and it can be view as a special case of a length 1 branched rollouts.

Formally, we can view it as executing a nonstationary policy which begins sampling from previous policy π_D, then at some specified time, we switch to unrolling the trajectory under model p and current policy π for k steps.

The returns now can be bouned as follows:

Theorem 2 *Given returns $\eta^{branch}[\pi]$ from the k-bounded rollout method,*

$$\eta[\pi] \geq \eta^{branch}[\pi] - 2r_{\max}[\frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^k+2}{(1-\gamma)}\epsilon_\pi + \frac{k}{1-\gamma}(\epsilon_m+2\epsilon_\pi)]$$

3.1.3 Model generalization in practice

With empirical experience, we can approximate $\epsilon_{m'}$:

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

Then we can arrive the following bound:

Theorem 3 *Under the k-branched rollout method, using model error under the updated policy $\epsilon_{m'} \geq \max_t E_{s\sim\pi_{D,t}}[D_{TV}(p(s',r|s,a)||p_\theta(s',r|s,a))]$, we have*

$$\eta[\pi] \geq \eta^{branch}[\pi] - 2r_{\max}[\frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^k+2}{(1-\gamma)}\epsilon_\pi + \frac{k}{1-\gamma}(\epsilon_m)]$$

Note: $k^* = \arg\min_k[\frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^k+2}{(1-\gamma)}\epsilon_\pi + \frac{k}{1-\gamma}(\epsilon_m)]$

3.2 Model-based policy optimization with deep reinforcement learning

A practical implementation of MBPO is presented in Algorithm 2. With two main differences: First is k-length rollouts from replay buffer steps, and the second one is a fixed number of policy update steps.

Algorithm 2 Model-Based Policy Optimization with Deep Reinforcement Learning

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

4 Conclusion

1. MBPO has some nice properties
   - has asymptotic performance rivaling the best model-free algorithms
   - learns substantially faster than prior model-free or model-based methods
   - is able to scale to long horizon tasks that often cause model-based methods to fail
2. In experiments, found that rollouts as short as a single step can provide pronounced benefits to policy optimization procedures.

References

S. Shalev-Shwartz and S. Ben-David. Understanding machine learning: From theory to algorithms. *Cambridge university press*, 2014.

R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *International Conference on Machine Learning*, 1990.