Model-based Policy Gradients with Entropy Exploration through Sampling

Samuel Stanton[1], Ke Alexander Wang[2], and Andrew Gordon Wilson[1]

[1]Department of Operations Research and Information Engineering, Cornell University
[2]Department of Computer Science, Cornell University

Summary

- We relax the analytic gradients constraint of PILCO [1] by using reparameterized trajectory samples to estimate expected costs.
- Our sampling approach allows gradient-based optimization of arbitrary differentiable cost functions without requiring complex analytic calculations.
- We introduce two different regularization terms to encourage policy exploration by maximizing entropy and minimizing reward variance.

Gaussian Process Dynamics Models

Given a deterministic environment with transition function f with continuous states $\mathbf{s} \in \mathbb{R}^D$ and actions $\mathbf{a} \in \mathbb{R}^F$, we model f using a Gaussian process

$$p(\mathbf{s}_{t+1} \mid \mathbf{x}_t) := \mathcal{GP}(\mathbf{s}_t, k(\mathbf{x}_t, \mathbf{x}_t'))$$

where $\mathbf{x}_t = (\mathbf{s}_t, \mathbf{a}_t) \in \mathbb{R}^{D+F}$ and each dimension of the GP is independent conditioned on $\mathbf{x}_t$.

Trajectory Sampling

Given a starting distribution $p(\mathbf{s}_0)$ and a deterministic policy $\pi(\cdot \mid \theta)$, we ancestral sample trajectories $\tau = (\mathbf{s}_0, \dots, \mathbf{s}_T)$ from the joint distribution $p(\tau) = p(\mathbf{s}_0)\, p(\mathbf{s}_1 \mid \mathbf{s}_0, \mathbf{a}_0) \dots p(\mathbf{s}_T \mid \mathbf{s}_{T-1}, \mathbf{a}_{T-1})$ where $\mathbf{a}_t = \pi(\mathbf{s}_t \mid \theta)$ and and Monte Carlo estimate the expected reward under dynamics model p using N trajectory samples

$$J(\theta) = \mathbb{E}_\tau[r(\tau)] \approx \Sigma_{i=1}^N \Sigma_{t=1}^T r(\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)}) \qquad (1)$$

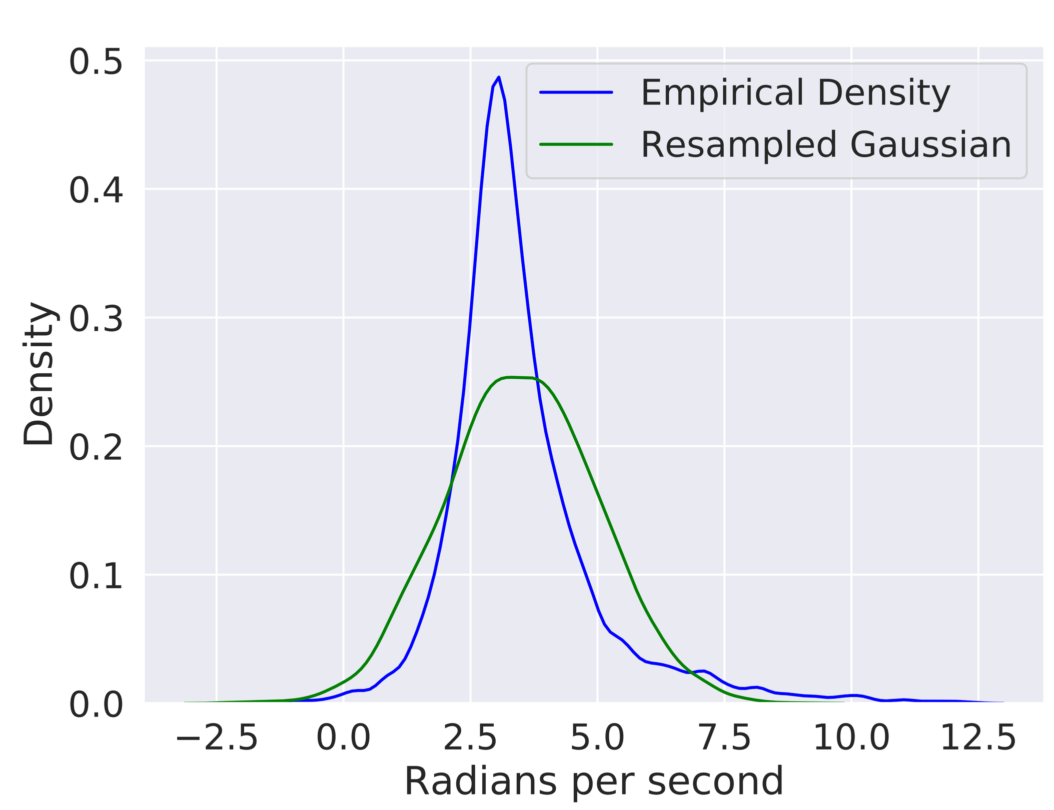

Figure 1: An example of the non-Gaussian marginal distribution of the pole velocity at $T = 30$ estimated using trajectory sampling and Gaussian resampling [2]

Algorithm

Input: Starting distribution $p(\mathbf{s}_0)$, initial policy $\pi(\cdot \mid \theta)$
Initialize dataset $\mathcal{D}$ by applying a random actions to environment
for episode $= 1, \dots, M$ **do**
  Learn GP dynamics model using $\mathcal{D}$
  Gradient-based optimization of $J(\theta)$ (1) by sampling trajectories
  Apply $\pi(\cdot \mid \theta)$ to environment, add observations to $\mathcal{D}$
end for
Output: $\pi(\cdot \mid \theta)$

Incentivizing Exploration

We explore the effects of two additional terms to the objective function to encourage exploration:

- Variance: maximize $J(\theta) - \frac{1}{N-1}\Sigma_{i=1}^N \left[\Sigma_{t=1}^T r(\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)}) - J(\theta)\right]^2$
- Entropy: maximize $J(\theta) + \frac{1}{N}\Sigma_{i=1}^N \Sigma_{t=1}^T \left[\frac{1}{2}\Sigma_{d=1}^D \log(\sigma_t^{(i)})_d^2\right]$

where $(\sigma_t^{(i)})_d^2$ is the variance of the d^{th} dimension of $p(\mathbf{s}_{t+1} \mid \mathbf{x}_t)$.

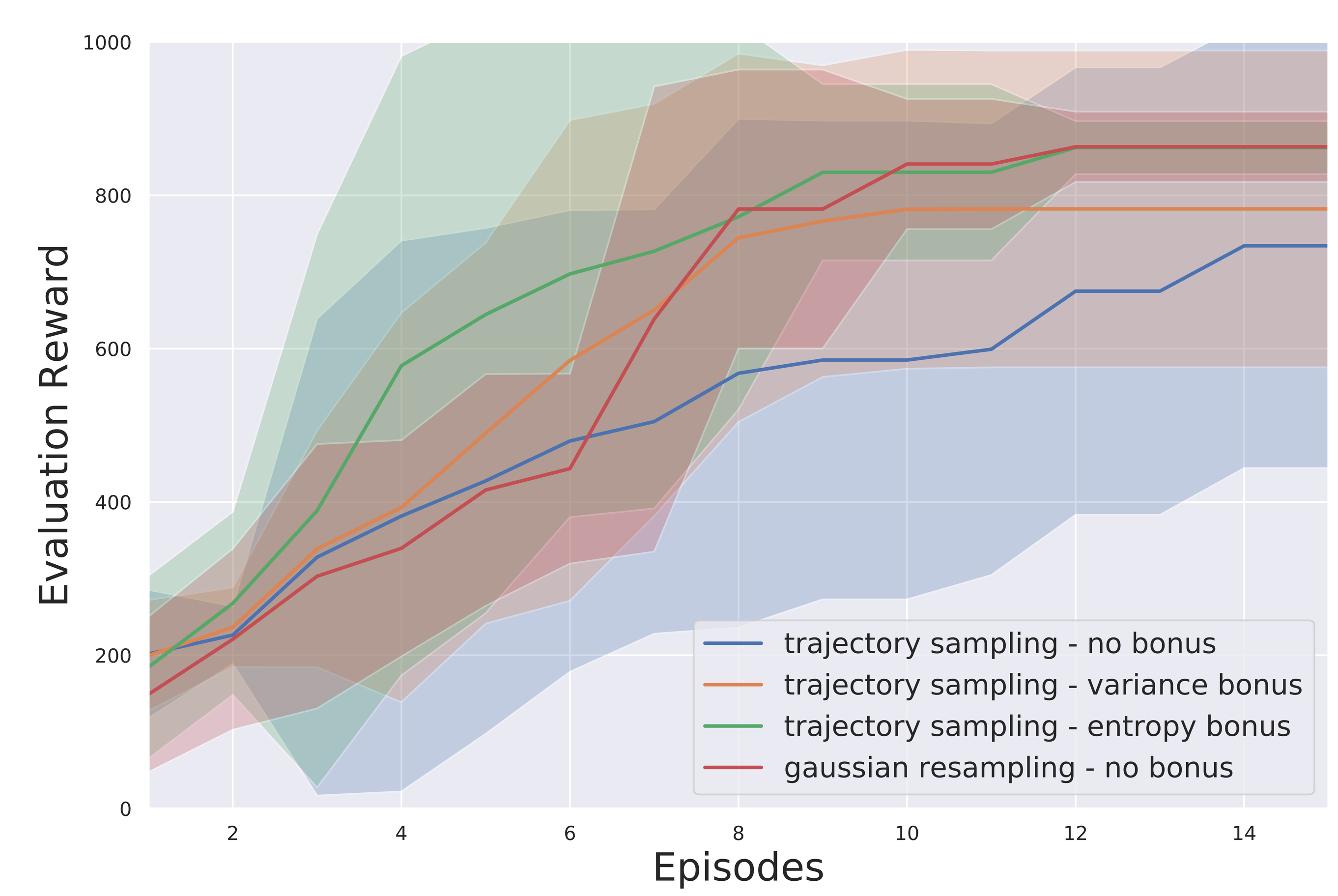

Figure 2: The reward curves for each method evaluated on the Deepmind Control Suite [3] cartpole swingup task averaged over 6 trials with 90% confidence intervals using $N = 256$ samples

Figure 2 shows that entropy and variance terms improve the reliability of trajectory sampling. Note that Gaussian resampling and trajectory sampling reproduce the sample efficiency from [1].

Advantages of Trajectory Sampling

- With trajectory sampling and automatic differentiation, we can use *any* differentiable cost function.
- We no longer need to make moment matching approximations or to manually derive

$$\frac{dJ}{d\theta} = \Sigma_{t=1}^T \frac{d\mathbb{E}_{\mathbf{x}_t}[r(\mathbf{x}_t)]}{d\theta}$$

$$\frac{d\mathbb{E}_{\mathbf{x}_t}[r(\mathbf{x}_t)]}{d\theta} = \frac{\partial \mathbb{E}_{\mathbf{x}_t}[r(\mathbf{x}_t)]}{\partial \mu_t}\frac{d\mu_t}{d\theta} + \frac{\partial \mathbb{E}_{\mathbf{x}_t}[r(\mathbf{x}_t)]}{\partial \Sigma_t}\frac{d\Sigma_t}{d\theta}$$

etc...

as done in [1]. Instead, we can instead simply sample from the model trajectory distribution with:

```
import torch
import gpytorch
...
def reward(states):
    ...
states = sampling_procedure(initial_dist)
obj = reward(states)
obj.backward()
```

References

[1] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, 2015.

[2] Paavo Parmas, Carl Edward Rasmussen, Jan Peters, and Kenji Doya. PIPPS: Flexible model-based policy search robust to the curse of chaos. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4065–4074, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[3] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Acknowledgements

We thank the reviewers for the ICML 2019 Workshop on Generative Modeling and Model-Based Reasoning for Robotics and AI for their helpful comments and Jacob R. Gardner for helpful discussions.

CornellEngineering
Operations Research and Information Engineering

Cornell CIS
COMPUTING AND INFORMATION SCIENCE