A K-Fold Method for Baseline Estimation in Policy Gradient Algorithms

Presented by Aleksander Beloi, Samsung SDS Research America

Abstract

Baseline helps reduce variance in unbiased policy gradient algorithms such as REINFORCE, VPG and TRPO. However, baseline fitting itself suffers from

- *Underfitting*: When the policy changes drastically between iterations
- *Overfitting*: When the current data is used to fit the baseline

We propose a K-fold method for baseline estimation that can be used as a tuning parameter to adjust the bias-variance trade-off.

Policy Gradient Algorithm

Parameter update loop:

1. Sample M trajectories $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$ using policy $\pi(a|s, \theta)$
2. Estimate the policy gradient $\mathbf{g}_\theta = \nabla_\theta \mathrm{E}_\tau[R(\tau) - b(\tau)]$ from sample data, where $R(\tau) = \sum_i \gamma^i r_i$ and $b(\tau)$ is a baseline function independent of θ.
3. Update the policy parameters using the policy gradients: $\theta \leftarrow \theta + \alpha \cdot \mathbf{g}_\theta$.

Baseline is used to **reduce variance** in gradient approximation step (2)

K-Fold Method

The K-fold method operates by breaking the data samples into K partitions. For each partition, a baseline is computed using data from all the other partitions.

- The fitting uses samples from the current policy, so we mitigate underfitting.
- We do not directly fit on the current partition's data samples, so we also mitigate overfitting.

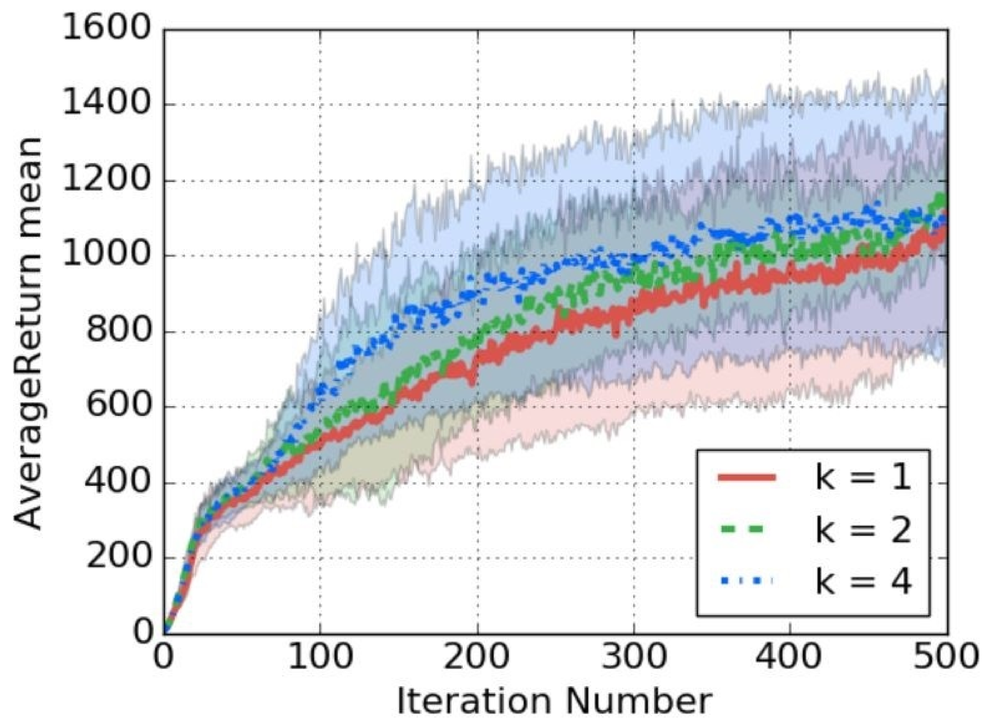
The K-Fold Algorithms

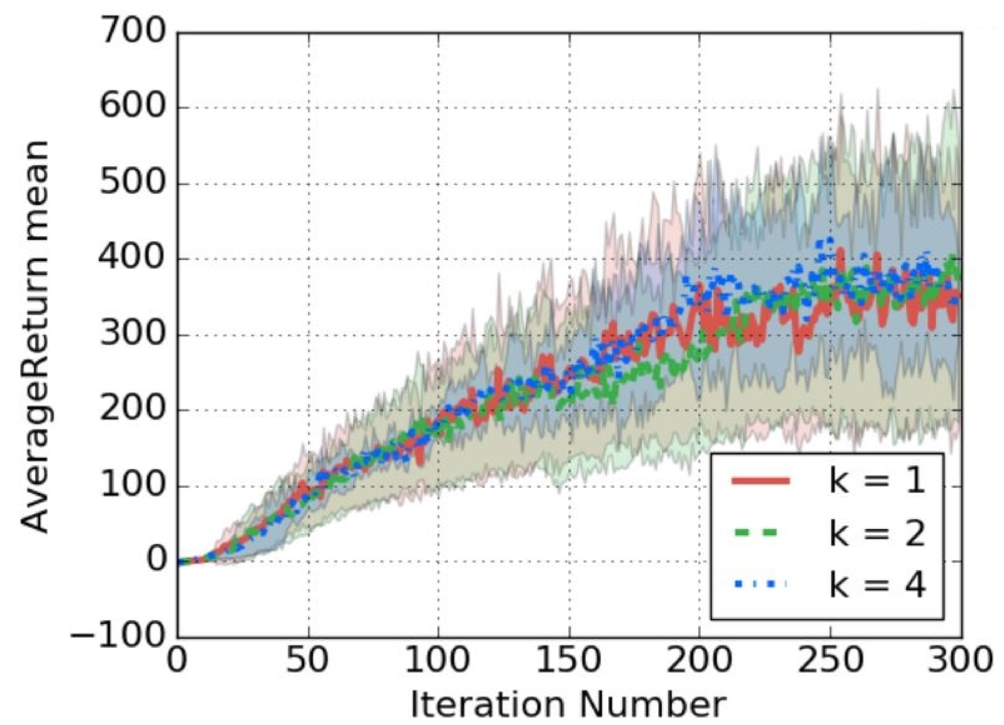
Method: We compute a baseline for each partition, use them to compute K different parameter updates and average all the parameters across the partitions.

Algorithm 1 Parameter-based K-Fold Baseline Estimation for Policy Optimization

Initialize: For iteration $i = 0$, initialize the policy parameter θ_0 randomly.
Iterate: Repeat for each iteration $i \in 1, 2, \ldots$ until convergence:

1: Sample N trajectories from policy $\pi(\cdot|\cdot, \theta_{i-1})$: $\tau_{j:1,\ldots,N}$.
2: Partition the N trajectories into $K < N$ disjoint subsets $S_1, \ldots, S_K$
3: **for** each subset S_k **do**
4: For each state $s_t \in \tau \in S_k$, compute discounted returns $R(\tau_{t:T}) = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$
5: Fit a baseline regression model $b_k^{(i-1)}(\cdot)$ for partition S_k using data from $\bigcup_{j \neq k} S_j$, i.e., with inputs $s_{t''} \in \tau' \in \bigcup_{j \neq k} S_j$ and outputs $R(\tau'_{t'':T})$.
6: Use policy optimization algorithm (such as the one to the left) with parameters θ_{i-1} and baseline $b_k^{(i-1)}(\cdot)$ to find optimized policy parameters θ_i^k.
7: **end for**
8: Update the policy parameters as the average of all the optimized policy parameters obtained, i.e. $\theta_i = \frac{1}{K} \sum_{k=1}^{K} \theta_i^k$.





(Left): Hopper with TRPO and a data size of 50000. (Right): Walker2D with TNPG and a data size of 5000.

Summary

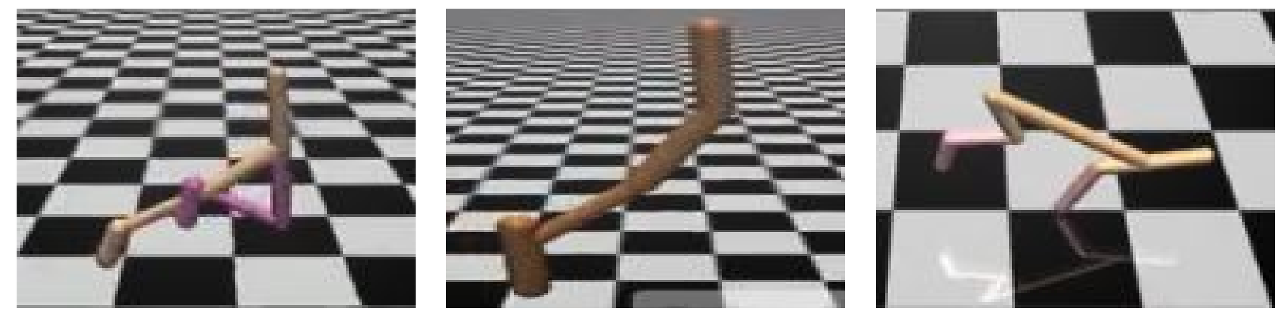
- The hyperparameter K is useful in achieving a good balance between overfitting and underfitting baseline.
- Future work will further this study for other environments, policy gradient algorithms, step sizes and batch sizes.

Experiment Setup

We use 5 random starting seeds and report the performance averaged over all the seeds.

- **Performance Metrics:** We define performance as the area under the average return curve.
- **Policy Network (Stochastic):** We use a feed-forward MLP network with 3 hidden layers of sizes 100, 50 and 25 with tanh nonlinearities after the first two hidden layers that maps states to the mean of a Gaussian distribution.
- **Baseline:** We use a Gaussian MLP for the baseline as well, with 2 hidden layers of size 32 each. The baseline is fitted using 10 ADAM steps.

Experimental Findings



Walker Hopper Half-Cheetah

Method 1 with TRPO and data size of 50,000.

Task	$K=1$	$K=2$	$K=4$
Walker	911.0 ± 681.0	$\mathbf{1015.7 \pm 327.3}$	938.7 ± 462.1
Hopper	727.7 ± 242.6	723.7 ± 190.5	$\mathbf{721.4 \pm 149.5}$
Cheetah	$\mathbf{1595.1 \pm 404.4}$	1528.5 ± 406.6	1383.8 ± 356.1

Method 2 with TRPO and data size of 50,000.

Task	$K=1$	$K=2$	$K=4$
Walker	911.0 ± 681.0	1035.0 ± 491.1	$\mathbf{1092.8 \pm 401.2}$
Hopper	727.7 ± 242.6	$\mathbf{786.0 \pm 171.1}$	847.7 ± 274.0
Cheetah	1595.1 ± 404.4	1664.1 ± 337.1	$\mathbf{1676.1 \pm 333.4}$

Method 2 with TNPG and data size of 5,000.

Task	$K=1$	$K=2$	$K=4$
Walker	299.4 ± 154.0	316.6 ± 164.6	$\mathbf{336.7 \pm 91.9}$
Hopper	331.4 ± 42.6	317.3 ± 29.5	$\mathbf{344.7 \pm 31.9}$
Cheetah	$\mathbf{609.5 \pm 215.3}$	445.9 ± 228.8	445.9 ± 181.9

SAMSUNG SAMSUNG SDS AMERICA

Nithyanand Kota[†], Abhishek Mishra[†], Sunil Srinivasa[†]
Samsung SDS Research America
[†]These authors contributed equally.
{n.kota, a2.mishra, s.srinivasa}@samsung.com

Xi (Peter) Chen, Pieter Abbeel
OpenAI
UC Berkeley, Department of EECS
{peter, pieter}@openai.com

OpenAI Berkeley