Statistical (Deep) Reinforcement Learning

Tianbing Xu

IDL, Baidu Research, USA

June 15, 2018

Outline (work done Feb. 2017 - Feb. 2018)

Tianbing Xu, Qiang Liu (Dartmouth), Jian Peng (UIUC)
Stochastic Variance Reduction for Policy Gradient Estimation
https://arxiv.org/pdf/1710.06034v2.pdf

Tianbing Xu
Variational Inference for Policy Gradient
https://arxiv.org/abs/1802.07833, Tech Report, 2018

Tianbing Xu, Qiang Liu (UT, Austin), Liang Zhao, Jian Peng (UIUC)
Learning to Explore via Meta-Policy Gradient
ICML 2018

Reinforcement Learning

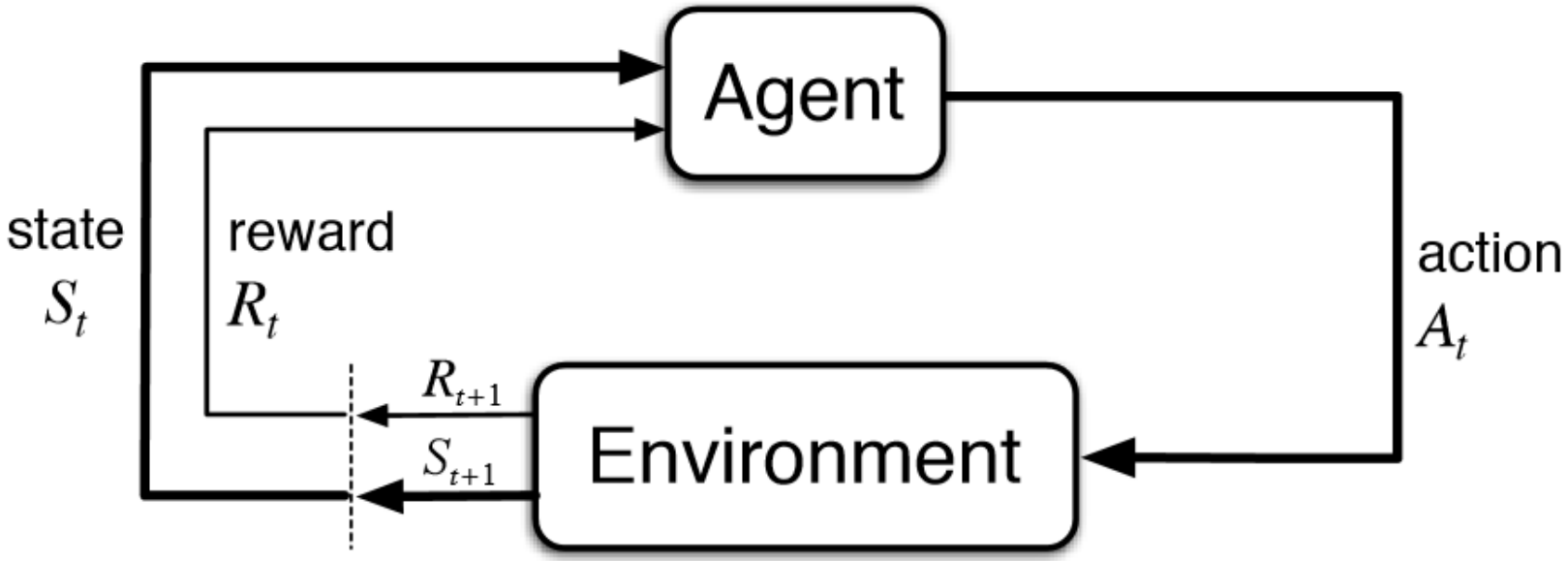


Figure: RL framework

Policy Optimization

Objective:

$$\max_{\theta} E[\sum_{t} r(s_t, a_t)|\pi_\theta]$$

Policy (parameterized as Neural Network):

$$\pi_\theta(a|s)$$

MDP (Markov Decision Process)

$$< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma >$$

- $\mathcal{S}$: state
- $\mathcal{A}$: action
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to R$
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$

Vanilla Policy Gradient - REINFORCE I

Objective (max the returns):

$$U(\theta) = E_{\pi_\theta}[R(\tau)] = \sum_{\tau} P_\theta(\tau) R(\tau)$$

Given trajectory (transition sequence) generated by exec policy π_θ:

$$\tau = \{s_0, a_0, r_0, ..., s_T, a_T, r_T\}$$

The Reward:

$$R(\tau) = \sum_{t} r(s_t, a_t)$$

Vanilla Policy Gradient - REINFORCE II

Policy Gradient:

$$\nabla_\theta U(\theta) = E_{\tau\sim\pi_\theta}[\nabla_\theta \log P_\theta(\tau) R(\tau)]$$

The derivation:

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \sum_\tau P_\theta(\tau) R(\tau) \\
&= \sum_\tau \frac{\nabla_\theta P_\theta(\tau)}{P_\theta(\tau)} P_\theta(\tau) R(\tau) \\
&= E_{\tau\sim\pi_\theta}[\nabla_\theta \log P_\theta(\tau) R(\tau)]
\end{aligned}$$

Learning to Explore via Meta-Policy Gradient

Tianbing Xu, Qiang Liu (UT, Austin), Liang Zhao, Jian Peng (UIUC)

Learning to Explore via Meta-Policy Gradient

ICML 2018

Video Result: ▸ Continuous Control Tasks

Learning to explore

Two-agents framework:

- Teacher (exploration policy π_e)
  generate exploration rollout for student, update exploration policy based on student's performance improvement
- Student (exploitation policy π)
  Objective is to improve its performance by learning from teacher's demonstrations

A view from Teacher-Student interactions

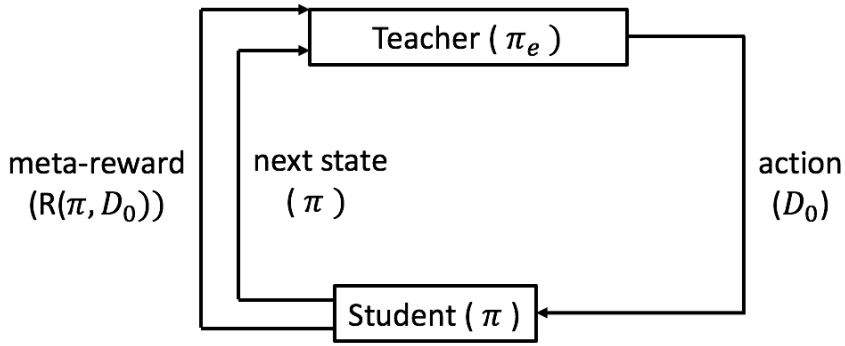


Figure: Illustrative view from interaction between teacher and student

Markov Decision Process for the Teacher and Student interactions

$$< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma >$$

- $\mathcal{S}$: state, exploitation policy π
- $\mathcal{A}$: action, rollout $D_0 = \{s_t, a_t, r_t, s'_t\}$ induced by exploration policy π_e
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, **Policy Updater**, e.g. DDPG
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow R$ **Meta-Reward**
  the performance improvement of student

Meta-Reward

The student performance improvement:

$$\mathcal{R}(\pi, D_0) = \mathcal{R}(\pi') - \mathcal{R}(\pi) \tag{1}$$

π' look-ahead policy of student

$$\pi' = DDPG(\pi, D_0)$$

$R(\pi)$ the cumulative reward of roll-out generated by policy π.

Learning to Explore with Meta-Policy Gradient

Teacher's Objective:

$$J(\pi_e) = E_{D_0 \sim \pi_e}[\mathcal{R}(\pi, D_0)] = E_{D_0 \sim \pi_e}[\mathcal{R}(DDPG(\pi, D_0)) - \mathcal{R}(\pi)]$$

Teacher's policy gradient:

$$\nabla_{\theta^{\pi_e}} J = E_{D_0 \sim \pi_e}[\mathcal{R}(\pi, D_0) \nabla_{\theta^{\pi_e}} \log P(D_0|\pi_e)] \tag{2}$$

here,

$$\nabla_{\theta^{\pi_e}} \log P(D_0|\pi_e) = \sum_t \nabla_{\theta^{\pi_e}} \log \pi_e(a_t|s_t)$$

Learning to Explore Algorithm

Algorithm 1 Learning to Explore

1: **for** iteration t **do**

2: Generate D_0 by executing teacher's policy π_e.

3: Update actor policy π to π' using DDPG based on D_0

4: Generate D_1 from π' and estimate the reward of π'. Calculate the meta reward: $\hat{\mathcal{R}}(\pi, D_0) = \hat{R}_{\pi'} - \hat{R}_{\pi}$.

5: Update Teacher's Policy π_e with meta policy gradient

$$\theta^{\pi_e} \leftarrow \theta^{\pi_e} + \eta \nabla_{\theta^{\pi_e}} \log \mathcal{P}(D_0|\pi_e)\hat{\mathcal{R}}(D_0)$$

6: Add both D_0 and D_1 into the Replay Buffer $B \leftarrow B \bigcup D_0 \bigcup D_1$.

7: Update π using DDPG based on Replay Buffer, that is, $\pi \leftarrow \mathrm{DDPG}(\pi,\ B)$. Compute the new $\hat{R}_{\pi}$.

8: **end for**

Experiments on Continuous Control Tasks

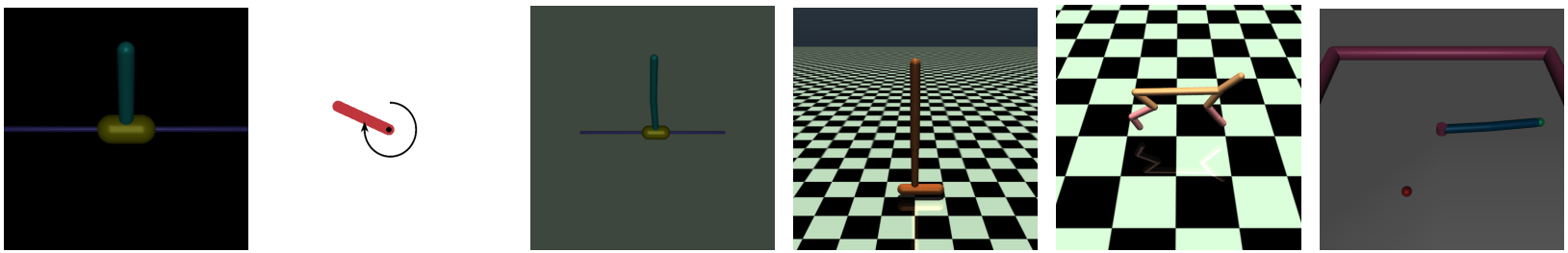


Figure: Illustrative screenshots of environments we experiment with Meta and DDPG

Meta Exploration Policy Explores Efficiently

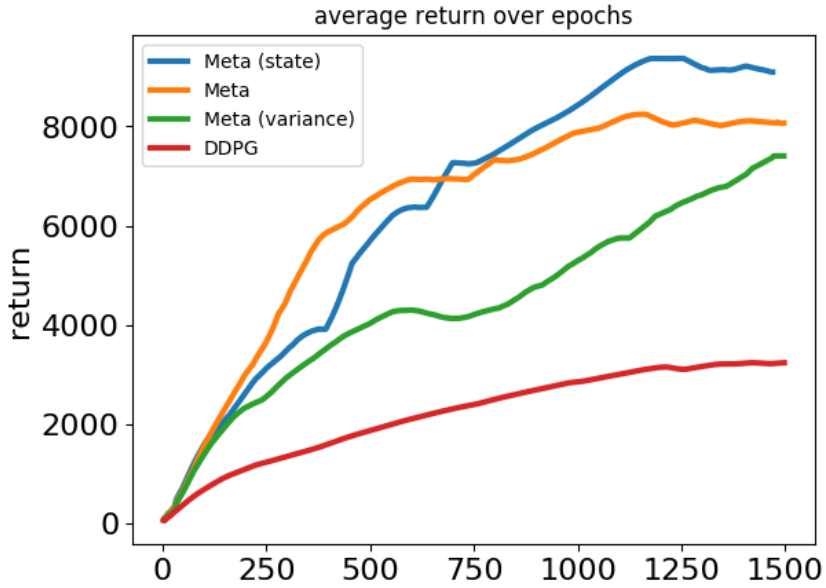


Figure: Comparison between meta exploration policies and DDPG

Meta:

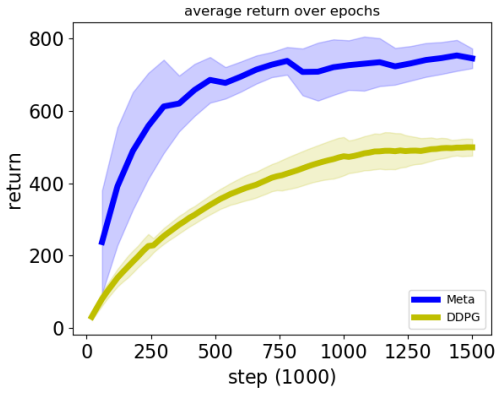
$$\pi_e \sim N(f(s, \theta^{\pi_e}), \Sigma)$$

Meta (State): adding more Q-function related features

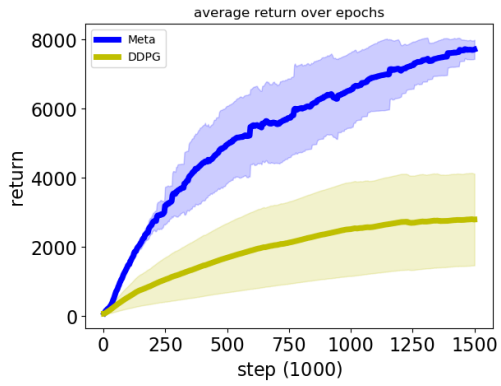
Meta Variance:

$$\pi_e \sim N(\mu(s, \theta^{\pi}), \Sigma)$$

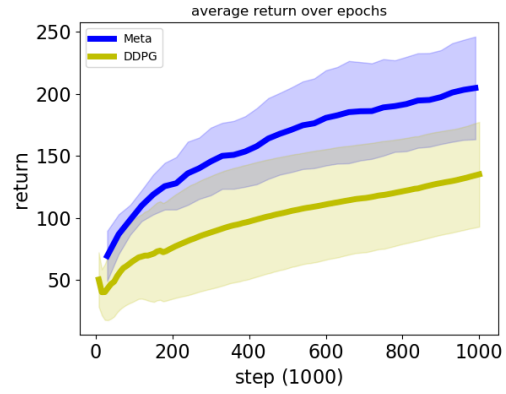
Sample Efficiency with Higher return



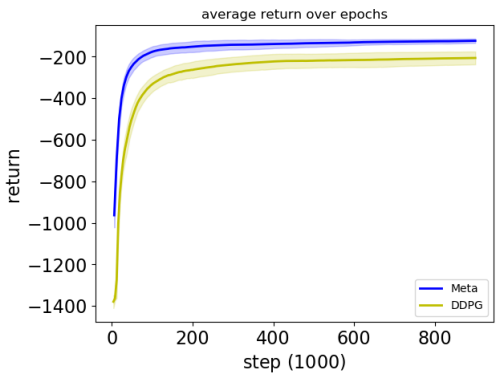
(a) IP



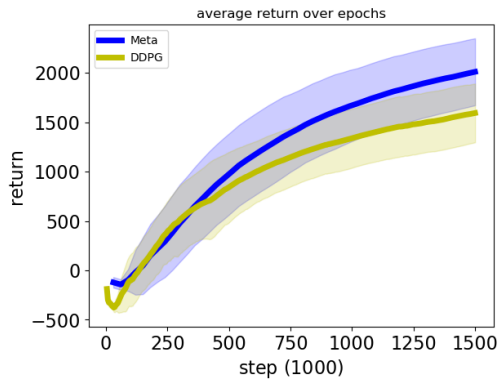
(b) IDP



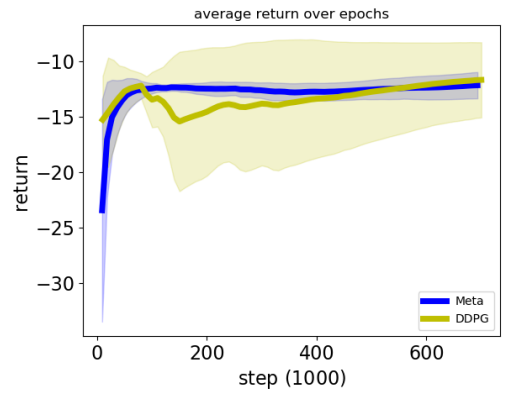
(c) Hopper



(d) Pendulum



(e) HalfCheetah



(f) Reacher

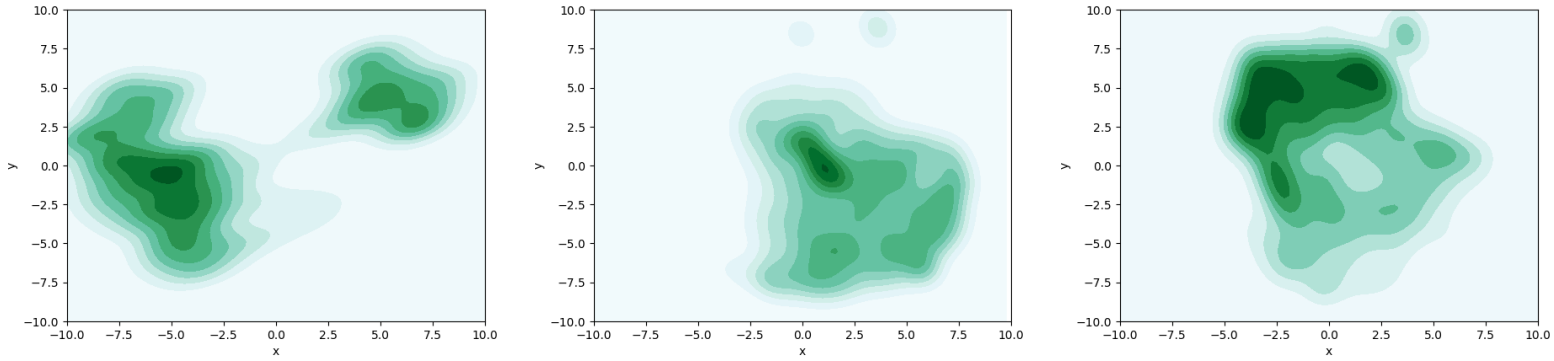
Figure: Performance Comparison of Meta and DDPG for Six Continuous Control Tasks.

Sample Efficiency with Higher Returns

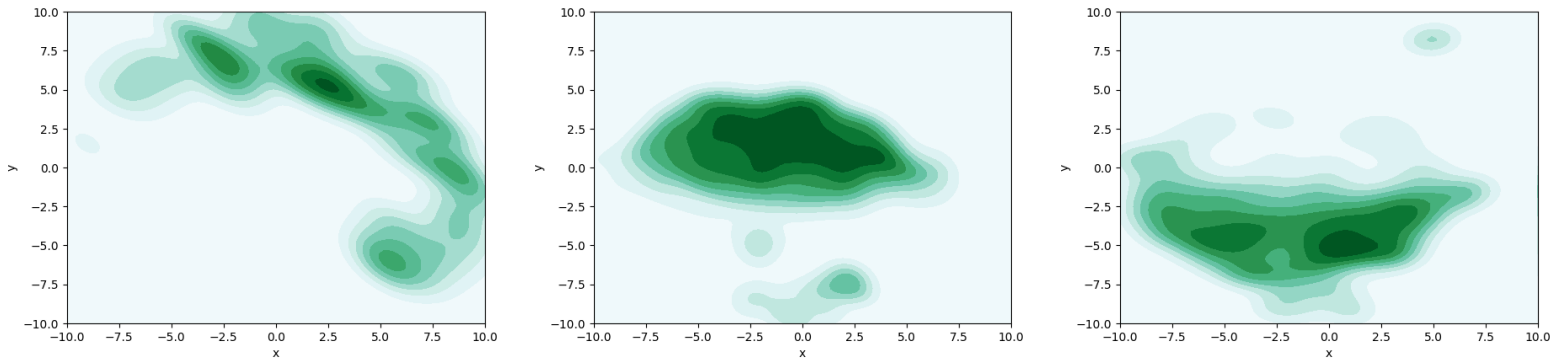
Table: Reward achieved in different environments

env-id	Meta	DDPG
InvertedDoublePendulum-v1	**7718 ± 277**	2795 ± 1325
InvertedPendulum-v1	**745 ± 27**	499 ± 23
Hopper-v1	**205 ± 41**	135 ± 42
Pendulum-v0	**-123 ± 10**	-206 ± 31
HalfCheetah-v1	**2011 ± 339**	1594 ± 298
Reacher-v1	-12.16 ± 1.19	**-11.67 ± 3.39**

Guided Exploration with Diverse and Adaptive Meta Policies



(a) Meta-Teacher (b) Meta-Student (c) DDPG



(d) Meta-Teacher (e) Meta-Student (f) DDPG

Figure: State Visitation Density Contours of Meta and DDPG in Early (the first row) and Late (the second row) Training Stages.

Conclusion

- Meta Exploration Policy Explores Efficiently with Sample Efficiency and Higher return
- Guided Exploration with Diverse and Adaptive Meta Policies
- *Global* Exploration in Different State Space compared to *Local* Exploration

For Further Reading


Andrychowicz, M., Denil, M., Gmez, S., and et al.
Learning to learn by gradient descent by gradient descent
NIPS 2016


Osband, I., Van Roy, B., and Zheng, W
Generalization and exploration via randomized value functions
ICML 2016


Finn, C., Abbeel, P., and Levine, S.
Model-agnostic metalearning for fast adaptation of deep networks.
ICML 2017

Stochastic Variance Reduction for Policy Gradient Estimation

Tianbing Xu, Qiang Liu (Dartmouth), Jian Peng (UIUC)
Stochastic Variance Reduction for Policy Gradient Estimation
https://arxiv.org/pdf/1710.06034v2.pdf

Video Result: ▸ Mujoco Tasks

Problem: High Variance

$$\nabla_\theta U(\theta) = E_{\tau \sim \pi_\theta}[\nabla_\theta \log P_\theta(\tau) R(\tau)]$$

$$\log P_\theta(\tau) = \sum_t \log \pi(a_t | s_t)$$

$$R(\tau) = \sum_t r(s_t, a_t)$$

- Vanilla Policy Gradient Estimation has very high variance
- long horizon, stochastic and noisy environment, stochastic policy
- Sample in-efficiency, need huge number of samples to learn the policy network

Stochastic Variance Reduction (Jonhson and Tong, NIPS 2013, Owen and Zhou, JASA 2000)

To find estimation:

$$\mu = E[f(X)]$$

Monte Carlo Estimation:

$$\hat{\mu} = \frac{1}{m} f(X_i)$$

Control Variate : $h(X)$ is close to $f(X)$

$$\nu = E[h(X] \quad \hat{\nu} = \frac{1}{m} h(X_i)$$

Stochastic Variance Reduction Estimation:

$$\hat{\mu}_{sv} = \nu + (\hat{\mu} - \hat{\nu})$$

Unbiased estimation:

$$E(\hat{\mu}_{sv}) = E[\hat{\mu}] = \mu$$

Policy Gradient Variance Reduction Estimation

Variance Reduction Policy Gradient Estimation:

$$\hat{\nabla}_{sv} U(\theta) = \hat{\nabla} U(\tilde{\theta}) + \left(\hat{\nabla}_m U(\theta) - \hat{\nabla}_m U(\tilde{\theta}) \right)$$

Policy Gradient Estimation:

$$\hat{\nabla}_m U(\theta) = \frac{1}{m} \sum_i U_i(\theta)$$

Control Variate $\tilde{\theta}$ (close to policy network parameter θ) :

$$\hat{\nabla}_m U(\tilde{\theta}) = \frac{1}{m} \sum_i U_i(\tilde{\theta})$$

Unbiased Estimation:

$$E[\hat{\nabla}_{sv}(\theta)] = E[\hat{\nabla}_m U(\theta)] = \nabla U(\theta)$$

Variance Reduction

Variance of Monte Carlo Estimation:

$$Var(\hat{\nabla}_m U(\theta)) = \frac{1}{m} Var(\nabla U(\theta))$$

Stochastic Variance Reduction:

$$Var(\hat{\nabla}_{sv} U(\theta)) \approx \frac{1}{m} Var\left(\nabla U(\theta) - \nabla U(\tilde{\theta})\right)$$

when $\tilde{\theta}$ is close to θ, the Variance is close to 0.
The derivation:

$$\begin{aligned} Var(\hat{\nabla}_{sv} U(\theta)) &= E[\left(\hat{\nabla} U(\tilde{\theta}) + \hat{\nabla}_m U(\theta) - \hat{\nabla}_m U(\tilde{\theta}) - \nabla U(\theta)\right)^2] \\ &\approx E[\left(\left(\hat{\nabla}_m U(\theta) - \hat{\nabla}_m U(\tilde{\theta})\right) - E\left(\hat{\nabla}_m U(\theta) - \hat{\nabla}_m U(\tilde{\theta})\right)\right)^2] \\ &= Var(\hat{\nabla}_m U(\theta) - \hat{\nabla}_m U(\tilde{\theta})) = \frac{1}{m} Var(\nabla U(\theta) - \nabla U(\tilde{\theta})) \end{aligned}$$

Stochastic Variance Reduction Algorithm

Algorithm 2 Stochastic Variance Reduction for Policy Optimization

for $\ell = 1$ to L **do**
  Initialize the parameter from control variate: $\theta^{\ell} = \tilde{\theta}^{\ell}$.
  Generate rollouts by exec the current policy $\pi_{\theta^{\ell}}$
  for $j = 1$ to J **do**
    Draw an uniformly random mini-batch I_j (with size m)
    Calculate the SVRG Policy Gradient estimation:

$$\hat{\nabla}_{sv} U(\theta^{\ell}) = \frac{1}{N} \sum_{i} \nabla U_i(\tilde{\theta}^{\ell}) + \frac{1}{m} \sum_{i \in I_j} \left(\nabla U_i(\theta^{\ell}) - \nabla U_i(\tilde{\theta}^{\ell}) \right).$$

    Update policy parameter with mini-batch I_j:

$$\theta^{\ell} \leftarrow \theta^{\ell} + \eta \hat{H}^{-1}(\theta^{\ell}) \hat{\nabla}_{sv} U(\theta^{\ell}).$$

  end for
  $\tilde{\theta}^{\ell+1} \leftarrow \theta^{\ell}$
end for

Connection to TRPO and Natural Policy Gradient

TRPO or NPO adopt Fisher information matrix with Monte Carlo Policy Gradient:

$$\theta \leftarrow \theta + \eta \hat{H}^{-1}(\theta)\hat{\nabla} U(\theta).$$

Additionally, we adopt Stochastic Variance Reduction Policy Gradient:

$$\theta \leftarrow \theta + \eta \hat{H}^{-1}(\theta)\hat{\nabla}_{sv} U(\theta).$$

To reduce the variance of policy gradient.

TRPO (Schulman et. al. ICML 2015)

$$\max_{\theta} L(\theta) = E_{\theta_{old}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} A^{\pi_{\theta\, old}}(a|s) \right]$$
$$s.t. \quad KL(\pi_{\theta_{old}}, \pi_\theta) \leq \delta$$

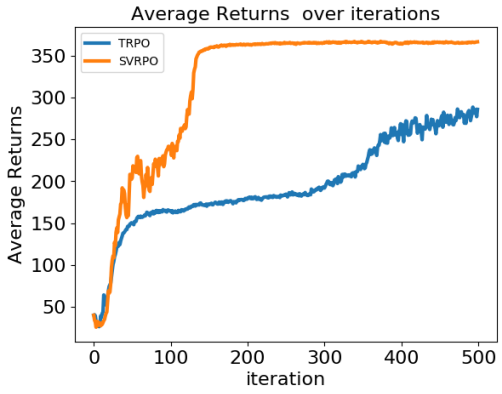
The surrogate objective $L(\theta)$ is the first order approximation to $U(\theta)$,

$$L(\theta) = U(\theta) \quad \nabla_\theta L(\theta) = \nabla_\theta U(\theta)$$

KL divergence is used as the trust region to stabilize the performance. Step size is bounded,

$$\eta \leq \sqrt{\frac{2\delta}{s' H s}}$$

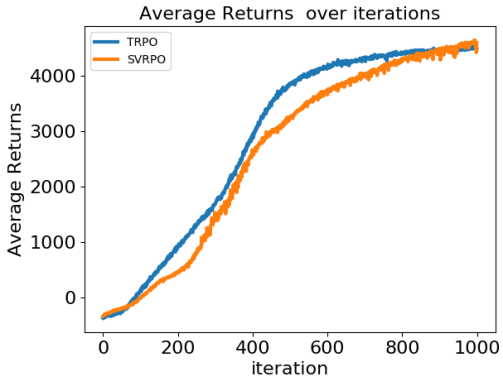
Results: Sample Efficiency and Higher Return



(a) Swimmer.

(b) Walker

(c) Hopper



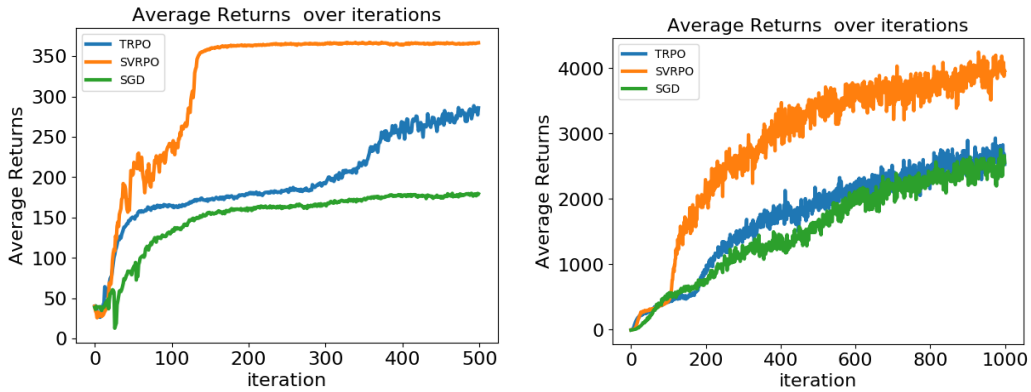
(d) Half-Cheetah

(e) Ant

(f) Humanoid

Figure: Performance Comparison of Variance Reduction and TRPO for six Mujoco Control Tasks.

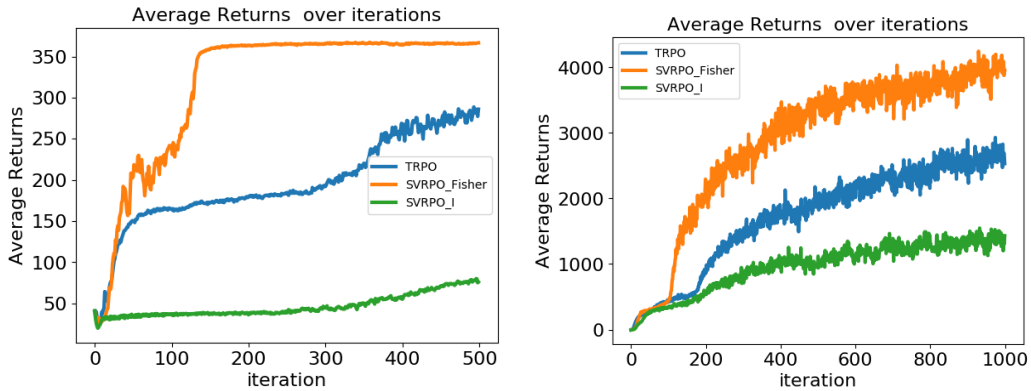
Results: Ablation Study (SVRG vs SGD)



(a) Swimmer. (b) Walker

Figure: Performance Advantage of Variance Reduction Policy gradient (SVRG vs vanilla SGD in our algorithm) with baseline TRPO for Swimmer and Walker.

Results: Ablation Study (Fisher Matrix)



(a) Swimmer. (b) Walker.

Figure: Fisher information matrix's accelerated convergence rates on our algorithm with baseline TRPO for Swimmer and Walker.

Running Video

- Swimmer
- Walker
- Hopper
- Half-Cheetah
- Ant

For Further Reading

Rie Johnson and Tong Zhang

Accelerating stochastic gradient descent using predictive variance reduction

NIPS 2013

Simon S. Du, Jianshu Chen, Lihong Li, Lin Xiao, Dengyong Zhou

Stochastic Variance Reduction Methods for Policy Evaluation

ICML 2017

Art Owen and Yi Zhou

Safe and Effective Importance Sampling

Journal of the American Statistical Association, 2000

For Further Reading

John Schulman and Sergey Levine and Philipp Moritz and Michael Jordan and Michael Jordan
Trust Region Policy Optimization
ICML 2015

Ronald J. Williams
simple statistical gradient following algorithms for connectionist reinforcement learning
Machine Learning 1992

Variational Inference for Policy Gradient

Tianbing Xu

Variational Inference for Policy Gradient

https://arxiv.org/abs/1802.07833, Tech Report, 2018

Bayesian Variational RL

Assume policy network parameter θ is a random variable.
The optimal distribution of θ:

$$p(\theta) \propto \exp\left(\frac{1}{\alpha} R(\theta)\right)$$

Expected cumulative return:

$$R(\theta) = E_{\pi_\theta}[\sum_t r(s_t, a_t)]$$

The Bayesian Formulation (Yang Liu, Qiang Liu, Jian Peng, UAI 2017)

To find $P(\theta)$ to maximize the following regularized expected return:

$$\tilde{R} = \int R(\theta) dp(\theta) + \alpha H(p)$$

Then, the optimal distribution of θ:

$$p(\theta) = \frac{1}{\mathcal{Z}} \exp\left(\frac{1}{\alpha} R(\theta)\right)$$

The derivation:

$$\begin{aligned}
\tilde{R} &= \alpha \int \left(\log \left(\frac{1}{p(\theta)} \right) + \frac{1}{\alpha} R(\theta) \right) dp(\theta) \\
&= \int \log \left(\frac{\exp(\frac{1}{\alpha} R(\theta))}{p(\theta)} \right) dp(\theta) \\
&= -KL(p(\theta) || \frac{1}{\mathcal{Z}} exp(\frac{1}{\alpha} R(\theta)))
\end{aligned}$$

Generating random variable by Transformation

How to generate an arbitrary random variable θ? We need an **invertible** (differential) transformation (e.g. Neural Network) h_ϕ

$$\xi \sim q_0$$
$$\theta = h_\phi(\xi)$$

Then, θ could induce a variational distribution $q_\phi(\theta)$ (next slide)

Variational Inference

To find a variational distribution of θ, $q_\phi(\theta)$ to match $p(\theta)$ s.t. :

$$\min_{\phi} KL(q_\phi||p)$$

Assume we could generate θ with a Neural Network mapping $h_\phi(\xi)$ from noise ξ. Then the Variational Inference is reduced to the problem of learning the transformation h_ϕ in an amortized fashion.

Difficulty

We try to learn the variational distribution $q_\phi(\theta)$ to match the optimal distribution of θ:

$$p(\theta) = \frac{1}{\mathcal{Z}} \exp\left(\frac{1}{\alpha} R(\theta)\right)$$

Difficulty: normalization factor $\mathcal{Z}$ (of the distribution to match) is **intractable**!

KL divergence with Transformation

$$KL(q_\phi||p) = -H(q_0) - E_{\xi\sim q_0}\left[\log p(h_\phi(\xi)) + \log\det(h_\phi(\xi)\xi)\right]$$

The derivation:

$$KL(q_\phi||p) = -H(q) - E_{q(\theta)}[\log p(\theta)]$$
$$= -H(q_0) - E_{\xi\sim q_0}\left[\log p(h_\phi(\xi)) + \log\det(h_\phi(\xi)\xi)\right]$$

Lemma

$$H(q) = H(q_0) + E_{\xi\sim q_0}\left[\log\det(h_\phi(\xi)\xi)\right]$$

Variational Inference by Learning from Transformation

Lemma (1)

Variational Policy Gradient:

$$
\begin{aligned}
-KL\phi &= E_{\xi\sim q_0}\left[\log p(h_\phi(\xi))\phi + \phi \log\det(h_\phi(\xi)\xi)\right] \\
&= E_{\xi\sim q_0}\left[\frac{1}{\alpha}R(\theta)\theta\theta\phi + \phi \log\det(h_\phi(\xi)\xi)\right] \qquad (3)
\end{aligned}
$$

Contribution: By-pass the normalization factor **intractable** problem. Now it is **tractable**.

- First part: maximize likelihood,
- Second: Repulsive force preventing collapsing towards MLE.

Adopt to Vanilla Policy Gradient

Vanilla Policy Gradient (insert into Eq.(3))

$$R(\theta)\theta = E_{\pi_\theta}\left[\sum_t \log \pi_\theta(a_t|s_t)\theta A(s_t, a_t)\right]$$

Advantage estimation (or GAE (Schulman 2015)):

$$A(s_t, a_t) = r(s_t, a_t) + V(s_t) - V(s_{t+1})$$

Continuous Control Example (Policy Network)

Gaussian MLP:

$$\pi_\theta(a|s) \propto \exp\left(-0.5 * (a - \theta_\phi(s,\xi))^T (a - \theta_\phi(s,\xi))\right)$$

Policy Network Generation Process:

$$\begin{aligned} \xi \sim N(0,\mathcal{I}), \zeta \sim N(0,\mathcal{I}) \\ \theta = h_\phi(s,\xi) = \mu_\phi(s) + \xi \cdot \sigma_\phi(s) \\ a = g_\theta(s,\zeta) = \theta(s,\xi) + \zeta \end{aligned}$$

Continous Control Example (Vanilla Policy Gradient)

Variational Policy Gradient (Lemma 1):

$$-KL\phi = E_{\xi \sim q_0}\left[\frac{1}{\alpha} R(\theta)\theta h_\phi(s,\xi)\phi + \sum_{i=0}^{d} \log \sigma_\phi^i(s)\phi\right]$$

Variational Policy Update:

$$\phi \leftarrow \phi - \eta \nabla_\phi KL$$

Connection to TRPO (Schulman et. al. ICML 2015)

The TRPO objective:

$$L(\theta) = E_{\theta_{old}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} A^{\pi_{\theta\, old}}(a|s) \right]$$

TRPO Variational Policy Gradient (Lemma 1),

$$-KL\phi = E_{\xi \sim q_0} \left[\frac{1}{\alpha} L(\theta)\theta\theta\phi + \phi \log\det(h_\phi(\xi)\xi) \right]$$

TRPO Variational Policy Update:

$$\phi \leftarrow \phi - \eta H^{-1}(\theta)\nabla_\phi KL$$

Connection to PPO (Schulman et. al. 2017)

PPO objective:

$$J_{ppo}(\theta) = E_{old}[\frac{\pi_\theta(a|s)}{\pi_{old}(a|s)} Q^\pi(a|s) - \lambda KL(\pi_{old}, \pi_\theta)]$$

PPO Variational Policy Gradient (Lemma 1),

$$-KL\phi = E_{\xi \sim q_0}\left[\frac{1}{\alpha} J_{ppo}(\theta)\theta\theta\phi + \phi \log\det(h_\phi(\xi)\xi)\right]$$

PPO Variational Policy Update:

$$\phi \leftarrow \phi - \eta \nabla_\phi KL$$

For Further Reading


Yihao Feng and Dilin Wang and Qiang Liu
Learning to Draw Samples with Amortized Stein Variational Gradient Descent
UAI 2017


Qiang Liu, Dilin Wang
Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm
NIPS 2016


Yang Liu, Prajit Ramachandran, Qiang Liu, Jian Peng
Stein Variational Policy Gradient
UAI 2017

For Further Reading

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, Pieter Abbeel

High-Dimensional Continuous Control Using Generalized Advantage Estimation

arXiv:1506.02438, 2015

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov

Proximal Policy Optimization Algorithms

arXiv:1707.06347, 2017