Deterministic Policy Gradient Algorithms

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Dann Wierstra, Martin Riedmiller
Google DeepMind

ICML 2014

Presented by Sanghyeon Lee

Motivation

Problem

Previous Policy Gradient methods updating based on action and state

➔ *Using Deterministic Policy instead of Stochastic Policy*

1. Propose a Deterministic Policy Algorithm
2. This paper show that Deterministic Policy is the special case of Stochastic Policy
3. DPG is more efficient than SPG
   - DPG has less computation
   - DPG has better performance than SPG especially high dim action space case

Method

Notation

$J(\pi) = E[r_1^\gamma | s, a\,; \pi]$: Average State value of all state

$\rho^\pi(s) = \lim_{t\to\infty} P(s_t = s | s_0, \pi_\theta)$: Stationary distribution of Markov chain for π_θ , $(\pi P = \pi)$

$V^\pi(s) = E[r_1^\gamma | S_1 = s; \pi]$: Value function, Expected total discounted reward

$Q^\pi(s, a) = E[r_1^\gamma | S_1 = s, A_1 = a; \pi]$ Action Value Function

$\mu_\theta(s)$: Deterministic Policy

$r_t^\gamma = \sum_{k=t}^{\infty} \gamma^{k-t} r(s_k, a_k)$: Total Discount Return

$\beta(a|s) \neq \pi_\theta(a|s)$: Another behavior policy (in off-policy setting)

Method

Background

1.1 Definition of Value function

- In continuing environments, we can't use discrete value for each state
- Instead of Value function, we can use the average value

$$J_{\mathrm{s_i}}(\theta) = \mathrm{V}_{\pi_\theta}(\mathrm{s_i}) \ \& \ \pi_\theta(\mathrm{s,a}) = \mathrm{P}[\mathrm{a}|\mathrm{s};\theta]$$

$$J(\pi) = E\left(r_1^\gamma \middle| \pi\right) = E_s[V_{\pi_\theta}(s)] = \sum_S \rho^\pi(s) V_{\pi_\theta}(s) = \sum_S \rho^\pi(s) \sum_a \pi_\theta(a|s) Q^\pi(s,a)$$

➔ Average reward per time-step

$$J(\pi_\theta) = \int_S \rho^\pi(s) \int_A \pi_\theta(s,a) r(s,a) dads = E_{s\sim\rho^\pi, a\sim\pi_\theta}[r(s,a)] \text{ (Integral form)}$$

1.2 Stochastic Policy Gradient Theorem

$$\nabla_\theta J(\pi_\theta) = \int_S \rho^\pi(s) \int_A \nabla_\theta \pi_\theta(a|s) \mathrm{Q}^\pi(s,a) dads = E_{s\sim\rho^\pi, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) \mathrm{Q}^\pi(s,a)]$$

-State Value function has summation form s.t state and action

Method

Background

1.3 Stochastic Actor-Critic Algorithm

1) Critic: Update action-value function

2) Actor: Improve policy by Gradient-Descent method

Action-Value Actor-critic

Simple actor-critic algorithm based on action-value critic

Input: π_θ, Q_w, step size $\alpha, \beta > 0$

Initialize s, θ, w at random.

For $t = 1, \ldots, T$

Sample $R_t \sim r(s,a)$ and the next state $s' \sim p(s'|s,a)$

Sample next action $a' \sim \pi_\theta(a'|s')$

$w \leftarrow w + \beta\big(R_t + \gamma Q_w(s',a') - Q_w(s,a)\big)\phi(s,a)$ Update critic

$\theta \leftarrow \theta + \alpha Q_w(s,a)\nabla_\theta \log \pi_\theta(a|s)$ Update actor

$a \leftarrow a', s \leftarrow s'$

* KAIST AI501 2019-2

Method

Background

1.4 Off-Policy Actor-Critic

Off−Policy : Behavior policy !=Improvement policy

$$J_\beta(\pi_\theta) = \int_S \rho^\beta(s)V^\pi(s)ds = \int_S \int_A \rho^\beta(s)\pi_\theta(s,a)dads$$

$$\nabla_\theta J_\beta(\pi_\theta) \approx \int_S \int_A \rho^\beta(s)\nabla\pi_\theta(a|s)Q^\pi(s,a)dads = \mathrm{E}_{s\sim\rho^\beta,a\sim\beta}[\frac{\pi_\theta(a|s)}{\beta_\theta(a|s)}\nabla_\theta\log\pi_\theta(a|s)Q^\pi(s,a)]$$

Pf) $\nabla_\theta J_\beta(\pi_\theta) = \int_S \int_A \rho^\beta(s)[\nabla\pi_\theta(a|s)Q^\pi(s,a) + \pi_\theta(a|s)\nabla Q^\pi(s,a)]dads$

$\approx \int_S \int_A \rho^\beta(s)\nabla\pi_\theta(a|s)Q^\pi(s,a)dads$ *Degris,2012b

- Importance Sampling

$$\mathrm{E}_{x\sim p} = \mathrm{E}_{x\sim q}[\frac{p}{q}f(x)]$$

Algorithm 1 The Off-PAC algorithm

Initialize the vectors $\mathbf{e}_v$, $\mathbf{e}_u$, and $\mathbf{w}$ to zero
Initialize the vectors $\mathbf{v}$ and $\mathbf{u}$ arbitrarily
Initialize the state s
For each step:
 Choose an action, a, according to $b(\cdot|s)$
 Observe resultant reward, r, and next state, s'
 $\delta \leftarrow r + \gamma(s')\mathbf{v}^\top\mathbf{x}_{s'} - \mathbf{v}^\top\mathbf{x}_s$
 $\rho \leftarrow \pi_\mathbf{u}(a|s)/b(a|s)$
 Update the critic (GTD(λ) algorithm):
  $\mathbf{e}_v \leftarrow \rho\left(\mathbf{x}_s + \gamma(s)\lambda\mathbf{e}_v\right)$
  $\mathbf{v} \leftarrow \mathbf{v} + \alpha_v\left[\delta\mathbf{e}_v - \gamma(s')(1-\lambda)(\mathbf{w}^\top\mathbf{e}_v)\mathbf{x}_s\right]$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha_w\left[\delta\mathbf{e}_v - (\mathbf{w}^\top\mathbf{x}_s)\mathbf{x}_s\right]$
 Update the actor:
  $\mathbf{e}_u \leftarrow \rho\left[\frac{\nabla_\mathbf{u}\pi_\mathbf{u}(a|s)}{\pi_\mathbf{u}(a|s)} + \gamma(s)\lambda\mathbf{e}_u\right]$
  $\mathbf{u} \leftarrow \mathbf{u} + \alpha_u\delta\mathbf{e}_u$
 $s \leftarrow s'$

Method

Gradient of Deterministic Polices

2.1 Action-Value Gradients

- Model free RL use greedy policy (Greedy policy: $\mu^{k+1}(s) = argmax_a Q^{\mu^k}(s,a)$)
- In the continuous action space, greedy policy improvement needs a global maximization at every step (It needs large cost)
- Instead of greedy policy, we can improve policy by maximize action-value function $Q^{\mu^k}(s,a)$

$$\theta^{k+1} = \theta^k + \alpha \mathrm{E}_{s\sim\rho^{\mu^k}}\left[\nabla_\theta Q^{\mu^k}\left(s,\mu_\theta(s)\right)\right] = \theta^k + \alpha \mathrm{E}_{s\sim\rho^{\mu^k}}[\nabla_\theta \mu_\theta(s) \nabla_\mathrm{a} \mathrm{Q}^{\mu^k}(\mathrm{s},\mathrm{a})|_{\mathrm{a}=\mu_\theta(s)}]$$ (Chain rule)

2.2 Deterministic Policy Gradient Theorem

$$J(\mu_\theta) = \int_S \rho^\mu(s) r\left(s,\mu_\theta(s)\right) ds = \mathrm{E}_{s\sim\rho^\mu}[\mathrm{r}\left(\mathrm{s},\mu_\theta(s)\right)]$$

$$\nabla_\theta J(\mu_\theta) = \int_S \rho^\mu(s) \nabla_\theta(\mathrm{s}) \nabla_a Q^\mu(s,a)\Big|_{a=\mu_\theta(s)} ds = \mathrm{E}_{s\sim\rho^\mu}[\nabla_\theta \mu_\theta(s) \nabla_\mathrm{a} \mathrm{Q}^\mu(\mathrm{s},\mathrm{a})\Big|_{\mathrm{a}=\mu_\theta(s)}]$$

- Deterministic policy only needs state distribution ρ^μ (SPG need ρ^μ and action distribution)

2.3 Limit of the Stochastic Policy Gradient = Deterministic Policy Gradient

$$\lim_{\sigma\downarrow 0} \nabla_\theta J\left(\pi_{\mu_\theta,\sigma}\right) = \nabla_\theta J(\mu_\theta)$$

➔Deterministic policy gradients are familiar of policy gradients

Regularity conditions A.1: $p(s'|s,a)$, $\nabla_a p(s'|s,a)$, $\mu_\theta(s)$, $\nabla_\theta \mu_\theta(s)$, $r(s,a)$, $\nabla_a r(s,a)$, $p_1(s)$ are continuous in all parameters and variables s, a, s' and x.

Regularity conditions A.2: there exists a b and L such that $\sup_s p_1(s) < b$, $\sup_{a,s,s'} p(s'|s,a) < b$, $\sup_{a,s} r(s,a) < b$, $\sup_{a,s,s'} ||\nabla_a p(s'|s,a)|| < L$, and $\sup_{a,s} ||\nabla_a r(s,a)|| < L$.

➔Lipschitz continuous& boundary condition

Method

Apply DPG – Deterministic Actor-Critic Algorithms

3.1 On-Policy Deterministic Actor-Critic

$$\delta_t = r_t + \gamma Q^w(s_{t+1}, a_{t+1}) - Q^w(s_t, a_t)$$

$$w_{t+1} = w_t + \alpha_w \delta_t \nabla_w Q^w(s_t, a_t) \quad (\nabla_w Q^w = w^T \nabla \phi(s,a))$$

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s_t) \, \nabla_a Q^w(s_t, a_t)|_{a=\mu_\theta(s)}$$

➔ Only Actor change

3.2 Off-Policy Deterministic Actor-Critic (OPDAC)

-We can avoid importance sampling in the actor and by using Q-learning, we can also avoid importance sampling in the critic

$$J_\beta(\mu_\theta) = \int_S \rho^\beta V^\mu(s) ds = \int_S \rho^\beta(s) Q^\mu(s, \mu_\theta(s)) ds$$

$$\nabla_\theta J_\beta(\mu_\theta) \approx \int_S \rho^\beta(s) \nabla_\theta \mu_\theta(a|s) Q^\mu(s,a) ds = \mathrm{E}_{s\sim\rho^\beta}[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}]$$

$$\delta_t = r_t + \gamma Q^w(s_{t+1}, a_{t+1}) - Q^w(s_t, a_t)$$

$$w_{t+1} = w_t + \alpha_w \delta_t \nabla_w Q^w(s_t, a_t)$$

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s_t) \, \nabla_a Q^w(s_t, a_t)|_{a=\mu_\theta(s)}$$

Method

Apply DPG – Deterministic Actor-Critic Algorithms

3.3 Compatible Function Approximation

- An approximator $Q^w(s,a)$ is not necessarily compatible with true gradient ; $Q^\mu(s,a)$

Theorem 3.

1) $\nabla_a Q^w(s,a)|_{a=\mu_\theta(s)} = \nabla_\theta \mu_\theta(s)^T w$ & 2) $\mathrm{MSE}(\theta, w) = E\left[\epsilon(s;\theta,w)^T \epsilon(s;\theta,w)\right]$ where

$$\epsilon(s;\theta,w) = \nabla_a Q^w(s,a)\Big|_{a=\mu_\theta(s)} - \nabla_a Q^\mu(s,a)\Big|_{a=\mu_\theta(s)}$$

Then, a function approximator $\boldsymbol{Q^w(s,a)}$ is compatible with a deterministic policy $\boldsymbol{\mu_\theta(s)}$,

$$\boldsymbol{\nabla_\theta J_\beta(\theta) = E[\nabla_\theta \mu_\theta(s) \nabla_a Q^w(s,a)\Big|_{a=\mu_\theta(s)}]}$$

Reduce Variance

- We can express approximator action-value function with baseline function independent of the action **a**

$$Q^w(s,a) = \left(a\text{-}\mu_\theta(s)\right)^T \nabla_\theta \mu_\theta(s)^T w + V^v(s), V^v \text{is independent with action}$$

(ex $V^v(s) = v^T \phi(s)$ for parameters v)

- *If* $a \approx \mu_\theta : Q^w(s,a) \approx V^v(s)$
- $A^w(s,a) = \phi(s,a)^T w,\ \phi(s,a) = \nabla_\theta \mu_\theta(s)\left(a - \mu_\theta(s)\right)$
- Advantage function $A^w(s,a) = Q_w(s,a) - V(s),\ ex)\ V(s) \approx \frac{1}{N}\sum Q^w(s_{i,t}, a_{i,t})$

Method

Apply DPG – Deterministic Actor-Critic Algorithms

Compatible off-policy deterministic actor critic with Q-learning (COPDAC-Q)

$$\delta_t = r_t + \gamma Q^w(s_{t+1}, \mu_\theta(s_{t+1})) - Q^w(s_t, a_t)$$
$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s_t) \left(\nabla_\theta \mu_\theta(s_t)^\top w_t\right)$$
$$w_{t+1} = w_t + \alpha_w \delta_t \phi(s_t, a_t)$$
$$v_{t+1} = v_t + \alpha_v \delta_t \phi(s_t)$$

Theorem3

$\nabla_\theta J_\beta(\theta) = E[\nabla_\theta \mu_\theta(s) \nabla_a Q^w(s,a)|_{a=\mu_\theta(s)},$

V is independent with action

Method

* Equivalent forms of policy gradient

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) G_t]$$ REINFORCE

$$= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) Q_{\boldsymbol{w}}(s,a)]$$ Q Actor-critic

$$= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) A_{\boldsymbol{w}}(s,a)]$$ Advantage Actor-critic

$$= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) \delta_t]$$ TD Actor-critic

$$\delta_{\boldsymbol{v}} = r + V_{\xi}(s') - V_{\xi}(s)$$

* KAIST AI501 2019-2

Experiments

1. Continuous Bandit

-Multidimensional bandit problem ($a \in R^m$),

$$Regret(\, difference\ between\ optimal\ policy\ rewards\ and\ choosen\ policy$$

$$Regret\big(-r(a)\big) = (a - a^*)^T C(a - a^*),$$

$$C\!: psd\ with\ eigenvalue \in \{0.1,1\}, a^* = [4, \ldots, 4]^T \in R^m)$$

- SAC-B: $\pi_{\theta,y}(*) \sim N(\theta, \exp(y))$, COPDAC-B: $\mu_\theta = \theta(\text{mean of Gaussian})$
- Critic is computed from each successive batch of 2m steps

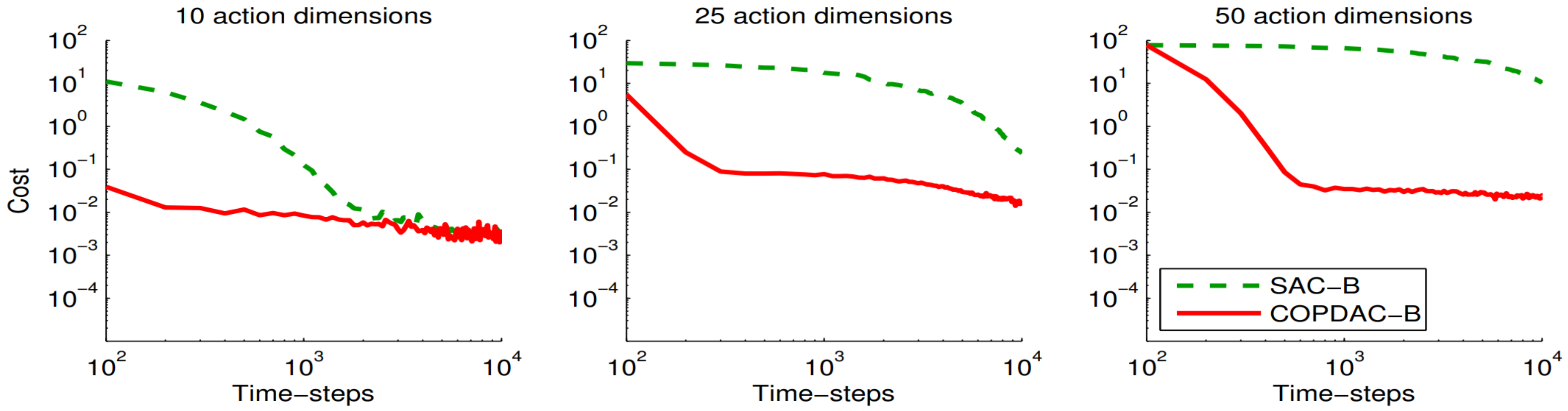
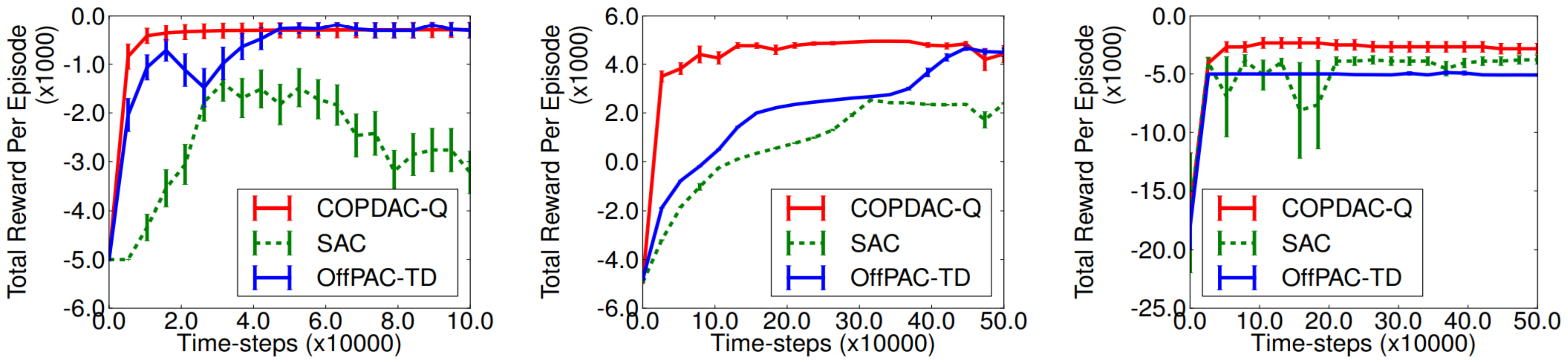


Figure 1. Comparison of stochastic actor-critic (SAC-B) and deterministic actor-critic (COPDAC-B) on the continuous bandit task.

Experiments

2. Continuous Reinforcement Learning (mountain car, pendulum and 2D puddle world)

- SAC-Q: $\pi_{\theta,y}(*) \sim N\left(\theta^T \phi(s), \exp\left(y^T \phi(s)\right)\right), V(s) = v^T \phi(s)$
- COPDAC-Q: $\mu_\theta(s) = \theta^T \phi(s)$, $\beta(* \mid s) \sim N\left(\theta^T \phi(s), \sigma_\beta^2\right), V(s) = v^T \phi(s)$
- OffPAC-TD: $\beta(* \mid s) \sim N\left(\theta^T \phi(s), \sigma_\beta^2\right)$, $\pi_{\theta,y}(*) \sim N\left(\theta^T \phi(s), \exp\left(y^T \phi(s)\right)\right)$, $V(s) = v^T \phi(s)$



(a) Mountain Car (b) Pendulum (c) 2D Puddle World

Figure 2. Comparison of stochastic on-policy actor-critic (SAC), stochastic off-policy actor-critic (OffPAC), and deterministic off-policy actor-critic (COPDAC) on continuous-action reinforcement learning. Each point is the average test performance of the mean policy.

Experiments

3. Octopus Arm

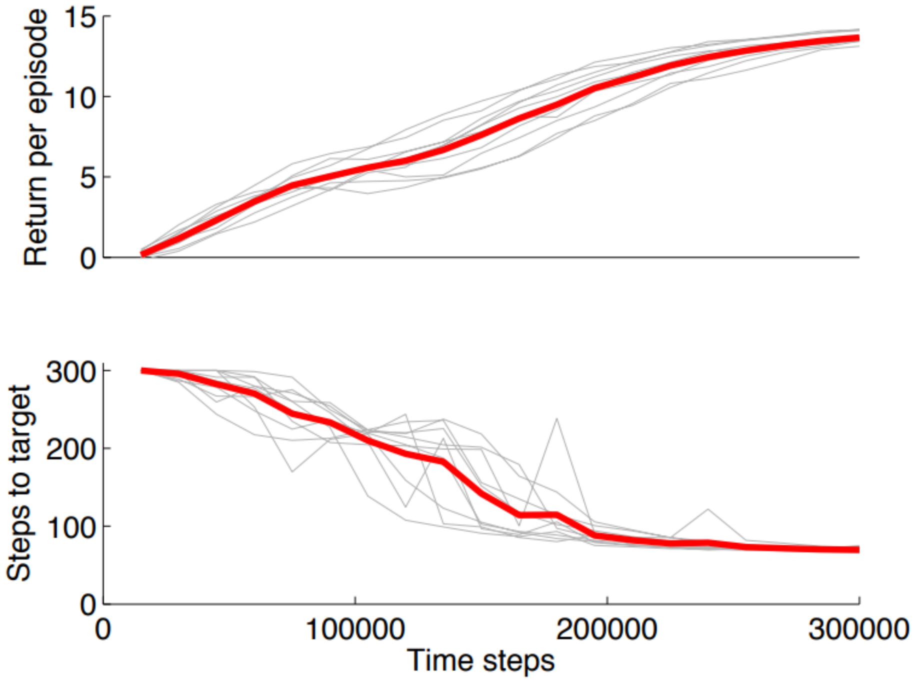


Figure 3. Ten runs of COPDAC on a 6-segment octopus arm with 20 action dimensions and 50 state dimensions; each point represents the return per episode (above) and the number of time-steps for the arm to reach the target (below).

Discussion

1. DPG is more efficient than SPG
   -DPG has less computation
   -DPG has better performance than SPG especially high dim action space case
2. But, It is not sutiable in the game which has stochastic optimal policy