# Intro: Difficulty and Advantage in Quantitate Investment

- Difficulty: noisy and unstable
- Advantage: option to say *I don’t know*
- Scheme 1: observation $\xrightarrow{ML\ models}$ prediction $\xrightarrow{strategy}$ trading signal
  - Problem: hard to train ML models (due to the difficulty - we did not make use of the advantage)
  - Possible solution: train ML model that yields not only prediction but also calibrated confidence (how to label confidence? i.e. predictable cases and unpredictable cases)
- Scheme 2: observation $\xrightarrow{RL}$ trading signal
  - Suffer from the difficulty at the same time benefit from the advantage

# Deterministic Policy Gradient

Silver, David, et al. "Deterministic policy gradient algorithms." *ICML*. 2014.

# Recap: Stochastic Policy Gradient Theorem

- Stochastic policy: $\pi_\theta(s, a) = \mathbb{P}[a|s, \theta]$
- RL as maximization: $J(\pi_\theta) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi_\theta(s, a) r(s, a) da\, ds$, where $\rho^\pi(s) = \int_{\mathcal{S}} \sum_{t=1}^{\infty} \gamma^{t-1} p_1(s') p(s' \rightarrow s, t, \pi) ds'$
- Policy gradient theorem (Sutton, 1999)*

$$\begin{aligned} \nabla_\theta J(\pi_\theta) &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a|s) Q^\pi(s, a) \mathrm{d}a \mathrm{d}s \\ &= \mathbb{E}_{s \sim \rho^\pi, a \sim \pi_\theta} [\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s, a)] \end{aligned}$$

- Key: how to estimate $Q^\pi(s, a)$?
    - REINFORCE (William, 1988, 1992): use MC (no bias, slow, high variance)



* Proof on Sutton's book *Reinforcement learning*

# Recap: Actor-critic and Compatible function

- Actor-critic:
  - actor $\rightarrow \pi_\theta(s, a)$; update along $\nabla_\theta J_\theta(\pi_\theta)$
  - critic $\rightarrow Q^w(s, a)$; update via any value function estimation algorithm
- Use function approximation to estimate $Q^w(s, a) \rightarrow Q^\pi(s, a)$
- Without bias: *compatible function approximator**
  - $Q^w(s, a) = \nabla_\theta \log \pi_\theta(a|s)^T \; w$
  - $w = \arg\min \epsilon^2(w) = \arg\min \mathbb{E}_{s \sim \rho^\pi, a \sim \pi_\theta}[\,(Q^w(s, a) - Q^\pi(s, a))^2]$



* Proof on Sutton, et al. *Policy gradient methods for reinforcement learning with function approximation*. NIPS 2000.

# Recap: Off-policy

- Off-policy: to explore more efficiently, especially when target policy is deterministic
- Off-policy policy gradient theorem*

$$J(\pi_\theta) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi_\theta(s,a) Q^\pi(s,a) da\, ds$$

$$\rightarrow J_\beta(\pi_\theta) = \int_{\mathcal{S}} \rho^\beta(s) \int_{\mathcal{A}} \pi_\theta(s,a) Q^\pi(s,a) da\, ds$$

$$\nabla_\theta J_\theta(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi}[\nabla_\theta \log \pi_\theta(s,a)\, Q^\pi(s,a)]$$

$$\rightarrow \nabla_\theta J_\beta(\pi_\theta) \approx \mathbb{E}_{s\sim\rho^\beta, a\sim\beta}\left[\frac{\pi_\theta(s,a)}{\beta_\theta(s,a)} \nabla_\theta \log \pi_\theta(s,a)\, Q^\pi(s,a)\right]$$



* Proof on Degris, Thomas, Martha White, and Richard S. Sutton. "Off-Policy Actor-Critic." (2012).

# From Stochastic to Deterministic

- All the above conclusion are based on stochastic policy gradient
- When target policy is deterministic, gradient computation doesn't need to sum over action space, thus faster.
- On-policy: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi}[\nabla_\theta \log \pi_\theta(s,a)\, Q^\pi(s,a)]$

$$\rightarrow \nabla_\theta J(\mu_\theta) = \mathbb{E}_{s\sim\rho^\mu}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)\Big|_{a=\mu_\theta(s)}\right]^*$$

- Off-policy: $\nabla_\theta J_\beta(\pi_\theta) \approx \mathbb{E}_{s\sim\rho^\beta, a\sim\beta}\left[\frac{\pi_\theta(s,a)}{\beta_\theta(s,a)} \nabla_\theta \log \pi_\theta(s,a)\, Q^\pi(s,a)\right]$

$$\rightarrow \nabla_\theta J_\beta(\mu_\theta) \approx \mathbb{E}_{s\sim\rho^\beta}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)\Big|_{a=\mu_\theta(s)}\right]$$



* Provided relevant functions are continuous

# From Stochastic to Deterministic

- Deterministic case is a limit of stochastic case

**Theorem 2.** *Consider a stochastic policy $\pi_{\mu_\theta,\sigma}$ such that $\pi_{\mu_\theta,\sigma}(a|s) = \nu_\sigma(\mu_\theta(s), a)$, where $\sigma$ is a parameter controlling the variance and $\nu_\sigma$ satisfy conditions B.1 and the MDP satisfies conditions A.1 and A.2. Then,*

$$\lim_{\sigma \downarrow 0} \nabla_\theta J(\pi_{\mu_\theta,\sigma}) = \nabla_\theta J(\mu_\theta)^{*} \quad (10)$$

*where on the l.h.s. the gradient is the standard stochastic policy gradient and on the r.h.s. the gradient is the deterministic policy gradient.*

* Conditions B.1: $\nu_\sigma$ is regular delta-approximation

# On-Policy Deterministic Actor-Critic

- (On-policy) deterministic policy gradient theorem

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

- Sarsa used to estimate action-value function

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

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

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

# Off-Policy Deterministic Actor-Critic

- (Off-policy) deterministic policy gradient theorem

$$\nabla_\theta J_\beta(\mu_\theta) \approx \mathbb{E}_{s\sim\rho^\beta}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)\Big|_{a=\mu_\theta(s)}\right]$$

- Q-learning used to estimate action-value function

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

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

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

# Compatible Function with Deterministic Policy

- Stochastic
  - $\nabla_\theta J(\pi_\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(s,a)\, Q^\pi(s,a)]$
  - $Q^w(s,a) = \nabla_\theta \log \pi_\theta(a|s)^T \; w$
  - $w = \arg\min \epsilon^2(w) = \arg\min \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}[\,(Q^w(s,a) - Q^\pi(s,a))^2]$
- Deterministic
  - $\nabla_\theta J(\mu_\theta) = \mathbb{E}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\right]$
  - $\nabla_a Q^w(s,a)|_{a=\mu_\theta(s)} = \nabla_\theta \mu_\theta(s)^T w$
  - $w = \arg\min \epsilon^2(w) = \arg\min \mathbb{E}[\,\left(\nabla_a Q^w(s,a)|_{a=\mu_\theta(s)} - \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\right)^2]$
- Compatible (= *no bias*): $\nabla_\theta J(\mu_\theta) = \mathbb{E}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^w(s,a)|_{a=\mu_\theta(s)}\right]$

# Specific Form of Compatible Function

- Deterministic
  - $\nabla_a Q^w(s,a)|_{a=\mu_\theta(s)} = \nabla_\theta \mu_\theta(s)^T w$
  - $w = \arg\min \epsilon^2(w) = \arg\min \mathbb{E}[\left(\nabla_a Q^w(s,a)|_{a=\mu_\theta(s)} - \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\right)^2]$
- $Q^w(s,a) = A^w(s,a) + V^v(s) = \phi(s,a)^T w + \phi(s)^T v = (a - \mu_\theta)^T \nabla_\theta \mu_\theta(s)^T w + \phi(s)^T v$
- This is defined to satisfy cond. 1. Use Q-learning to approximately satisfy cond. 2

# Compatible Off-Policy Deterministic Actor-Critic (COPDAC)

- $Q^w(s,a) = \phi(s,a)^T w + \phi(s)^T v = (a - \mu_\theta)^T \nabla_\theta \mu_\theta(s)^T w + \phi(s)^T v$

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

- Gradient Q-learning to prevent diverge (COPDAC-GQ)*

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

* For detail see Sutton's book chapter 11.7

# Experiment: Continuous Bandit Task

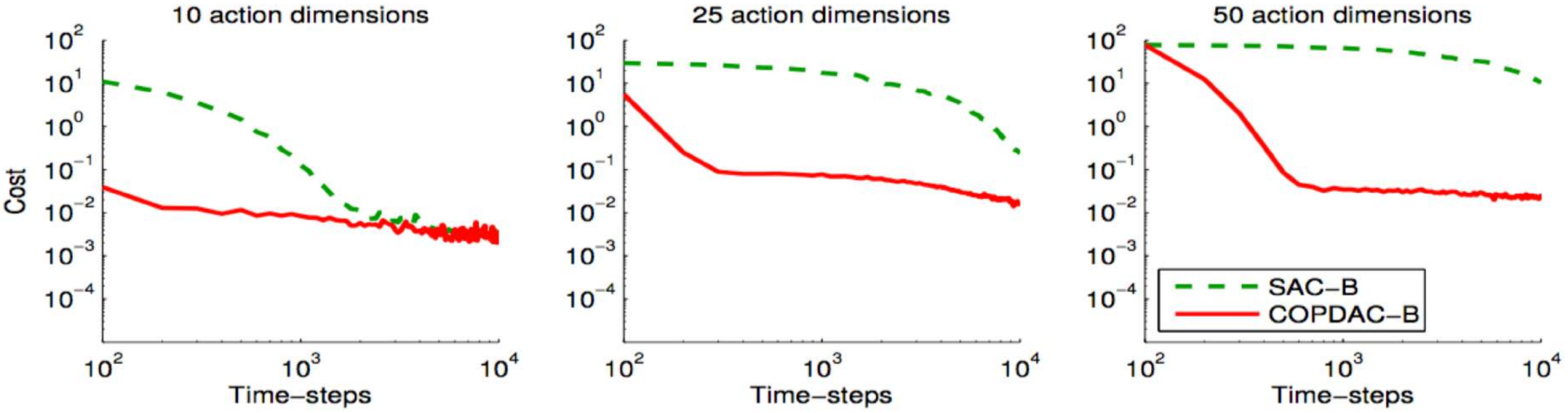
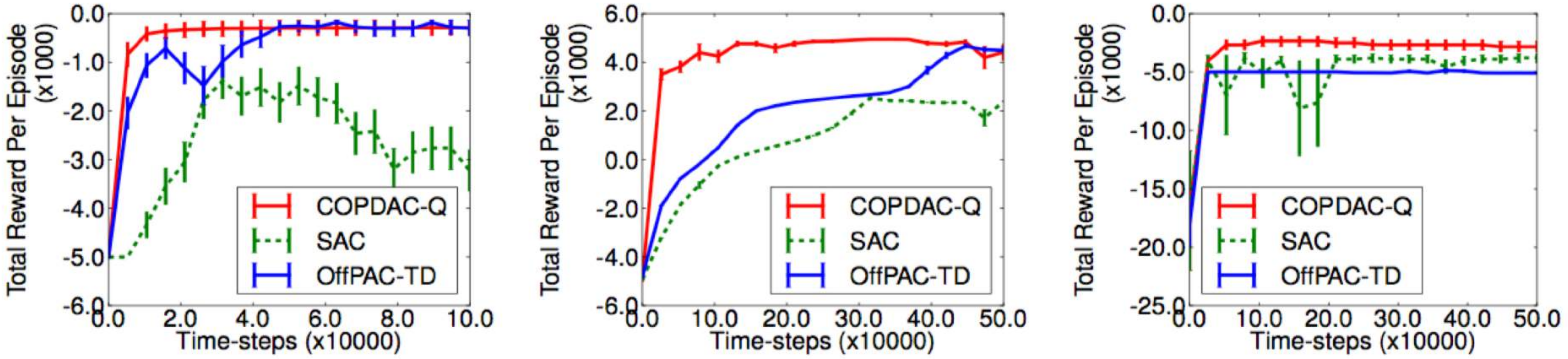



*Figure 1.* Comparison of stochastic actor-critic (SAC-B) and deterministic actor-critic (COPDAC-B) on the continuous bandit task.

SAC-B (stochastic actor-critic): $\pi_{\theta,y}(\cdot) \sim \mathcal{N}(\theta, \exp(y))$

COPDAC-B: $\beta(\cdot) \sim \mathcal{N}\left(\mu_\theta, \sigma_\beta^2\right), \mu_\theta = \theta$

# Experiment: Continuous RL




(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.

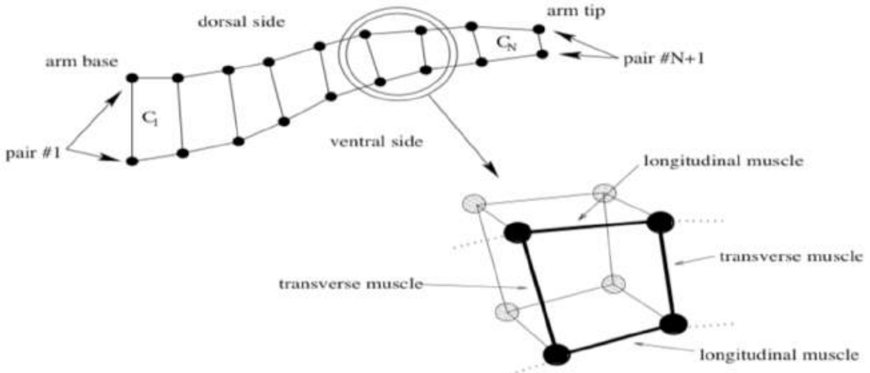
SAC (stochastic actor-critic): $\pi_{\theta,y}(s,\cdot) \sim \mathcal{N}(\theta^T \phi(s), \exp(y^T \phi(s)))$

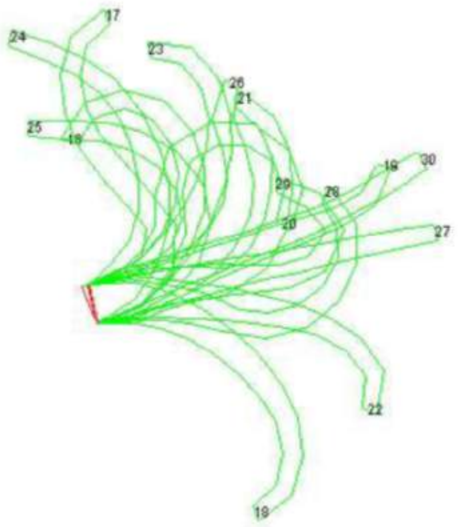
OffPAC (off-policy actor-critic): explore as $\beta(\cdot\,|s)$, learn a stochastic policy $\pi_{\theta,y}(s,\cdot)$ as SAC

COPDAC: $\beta(\cdot\,|s) \sim \mathcal{N}\left(\mu_\theta(s), \sigma_\beta^2\right), \mu_\theta(s) = \theta^T \phi(s)$

# Experiment: Octopus Arm

- Task: 6 segments attached to a rotating base; strike the target with any part of the arm
- 50 continuous state var.; 20 action var.



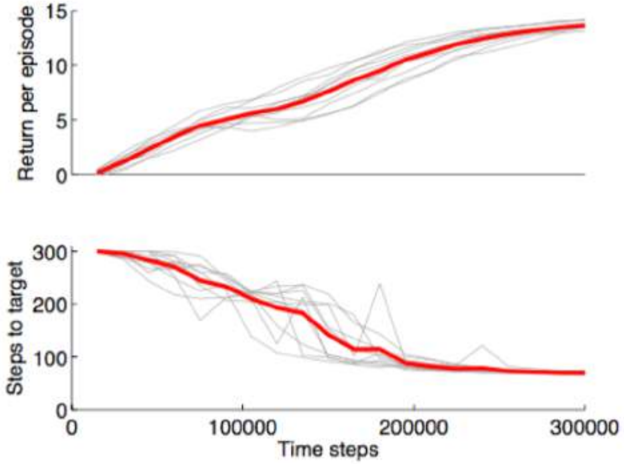



*Figure 3.* Ten runs of COPDAC on a 6-segment octopus arm with 20 action dimensions and 50 state dimensions; each point represents the return per episode (above) and the number of time-steps for the arm to reach the target (below).

# Deep Deterministic Policy Gradient (DDPG)

Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." *ICML*. 2016.

# Recap: Deep Q-learning

- Q-learning:
  - $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s,a)]$
  - Policy $\pi$: $a \leftarrow \max_{a'} Q(s, a')$
- Learn NN parameters for Q-learning in stable and robust way
  - Replay buffer to minimize correlation between samples
  - Target Q network to give consistent target
- When action space is high-dimensional continuous space?
  - $\rightarrow$ Deterministic policy gradient

# DDPG

**Algorithm 1** DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $R$
**for** episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $s_1$
  **for** t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
    Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
    Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  **end for**
**end for**

Ornstein-Uhlenbeck process

Replay buffer

Critic network training (with batchnorm)

Actor network training (with batchnorm)

Target Q network and target policy network

# Experiment

## Input: 64x64(pixels)x3(rgb)x3(action repeat)

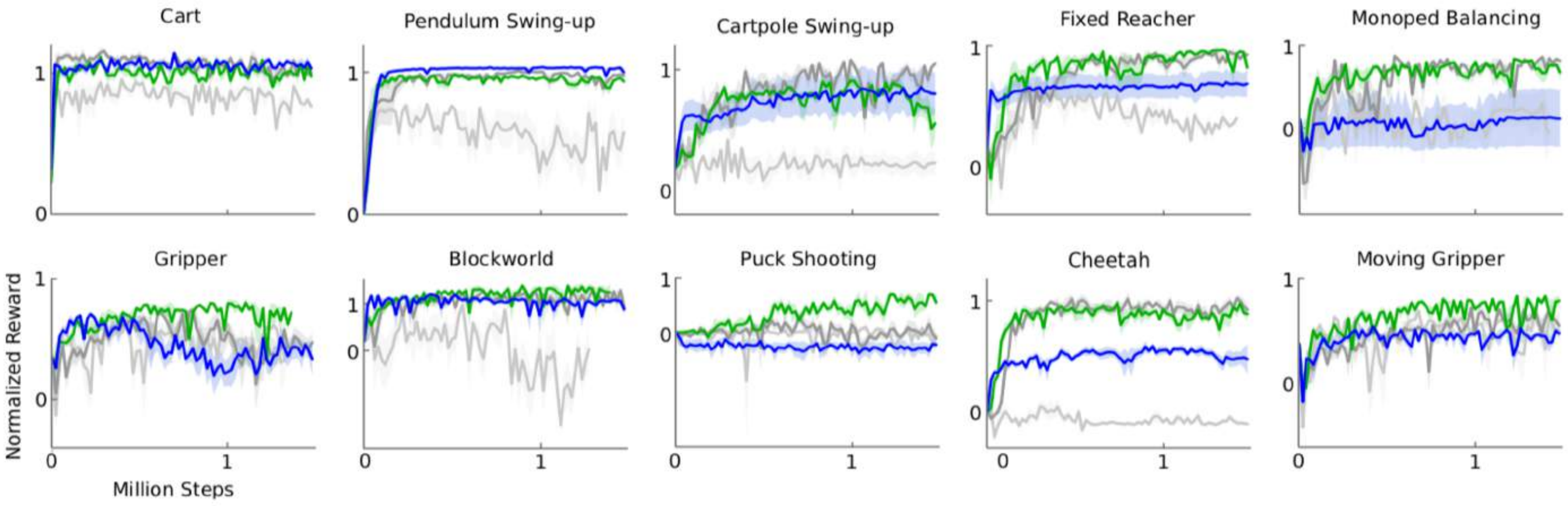



Figure 2: Performance curves for a selection of domains using variants of DPG: original DPG algorithm (minibatch NFQCA) with batch normalization (light grey), with target network (dark grey), with target networks and batch normalization (green), with target networks from pixel-only inputs (blue). Target networks are crucial.

# Experiment

Table 1: Performance after training across all environments for at most 2.5 million steps. We report both the average and best observed (across 5 runs). All scores, except Torcs, are normalized so that a random agent receives 0 and a planning algorithm 1; for Torcs we present the raw reward score. We include results from the DDPG algorithn in the low-dimensional ($lowd$) version of the environment and high-dimensional ($pix$). For comparision we also include results from the original DPG algorithm with a replay buffer and batch normalization ($cntrl$).

| environment | $R_{av,lowd}$ | $R_{best,lowd}$ | $R_{av,pix}$ | $R_{best,pix}$ | $R_{av,cntrl}$ | $R_{best,cntrl}$ |
|---|---|---|---|---|---|---|
| blockworld1 | 1.156 | 1.511 | 0.466 | 1.299 | -0.080 | 1.260 |
| blockworld3da | 0.340 | 0.705 | 0.889 | 2.225 | -0.139 | 0.658 |
| canada | 0.303 | 1.735 | 0.176 | 0.688 | 0.125 | 1.157 |
| canada2d | 0.400 | 0.978 | -0.285 | 0.119 | -0.045 | 0.701 |
| cart | 0.938 | 1.336 | 1.096 | 1.258 | 0.343 | 1.216 |
| cartpole | 0.844 | 1.115 | 0.482 | 1.138 | 0.244 | 0.755 |
| cartpoleBalance | 0.951 | 1.000 | 0.335 | 0.996 | -0.468 | 0.528 |
| cartpoleParallelDouble | 0.549 | 0.900 | 0.188 | 0.323 | 0.197 | 0.572 |
| cartpoleSerialDouble | 0.272 | 0.719 | 0.195 | 0.642 | 0.143 | 0.701 |
| cartpoleSerialTriple | 0.736 | 0.946 | 0.412 | 0.427 | 0.583 | 0.942 |
| cheetah | 0.903 | 1.206 | 0.457 | 0.792 | -0.008 | 0.425 |
| fixedReacher | 0.849 | 1.021 | 0.693 | 0.981 | 0.259 | 0.927 |
| fixedReacherDouble | 0.924 | 0.996 | 0.872 | 0.943 | 0.290 | 0.995 |
| fixedReacherSingle | 0.954 | 1.000 | 0.827 | 0.995 | 0.620 | 0.999 |
| gripper | 0.655 | 0.972 | 0.406 | 0.790 | 0.461 | 0.816 |
| gripperRandom | 0.618 | 0.937 | 0.082 | 0.791 | 0.557 | 0.808 |
| hardCheetah | 1.311 | 1.990 | 1.204 | 1.431 | -0.031 | 1.411 |
| hopper | 0.676 | 0.936 | 0.112 | 0.924 | 0.078 | 0.917 |
| hyq | 0.416 | 0.722 | 0.234 | 0.672 | 0.198 | 0.618 |
| movingGripper | 0.474 | 0.936 | 0.480 | 0.644 | 0.416 | 0.805 |
| pendulum | 0.946 | 1.021 | 0.663 | 1.055 | 0.099 | 0.951 |
| reacher | 0.720 | 0.987 | 0.194 | 0.878 | 0.231 | 0.953 |
| reacher3daFixedTarget | 0.585 | 0.943 | 0.453 | 0.922 | 0.204 | 0.631 |
| reacher3daRandomTarget | 0.467 | 0.739 | 0.374 | 0.735 | -0.046 | 0.158 |
| reacherSingle | 0.981 | 1.102 | 1.000 | 1.083 | 1.010 | 1.083 |
| walker2d | 0.705 | 1.573 | 0.944 | 1.476 | 0.393 | 1.397 |
| torcs | -393.385 | 1840.036 | -401.911 | 1876.284 | -911.034 | 1961.600 |