Policy gradient

Konpat Preechakul
Chulalongkorn University
October 2019

Recap overview

Model-free	Model-based
Environment is a black box	We know environment (transitions, rewards)

Value-based	Policy-based
We learn value. Use value to improve policy greedily	We directly learn policy (from some value)

On-policy	Off-policy
Experience comes from target policy (interactive experience)	Experience comes from behavior policy (observative experience)

Recap Value-based

Value function approximation

- Monte Carlo
  - Value error
- Temporal difference
  - Semi gradient
    - Poor convergence
    - Tricks
  - Full gradient
    - Better convergence
    - Might not good in practice

Today topic is policy-based

- Model the policy directly (not from the value function)

- Value function is used as a guide

Motivation of policy-based

- Continuous action space
- Optimal policy might not be “deterministic”
- Value-based RL might not give a smooth improvement of the policy
- Convergence problems in value based RL

Modeling a policy

- Stochastic policy

- Deterministic policy

How to improve the policy?

- We have neural nets, we want to optimize it with SGD
- What is the objective?
- What is the gradient?

Policy gradient

Policy gradient (PG)

- Objective function

$$J(\theta) = \sum_{s} P_{s_0}(s) V^{\pi}(s)$$

- Policy gradient

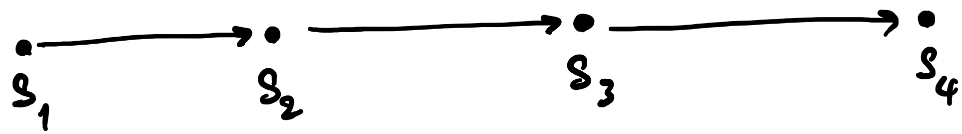
$$\nabla J(\theta) = \sum_{s} d^{\pi}(s) \sum_{a} Q^{\pi}(s,a) \nabla_{\theta} \pi_{\theta}(a|s)$$

Sutton, Richard S., David Mcallester, Satinder Singh, and Yishay Mansour. 1999. "Policy Gradient Methods for Reinforcement Learning with Function Approximation." *In Advances in Neural Information Processing Systems 12*, 1057–63.

Discounted state distribution

- What does it mean?

$d^{\pi}(s)$



- We care less of far away states
- Why?

Policy gradient (PG)

- Another form (REINFORCE; likelihood ratio)

$$\nabla J(\theta) = \mathbb{E}_{s,a}\left[Q^{\pi}(s,a)\nabla_{\theta}\log\pi_{\theta}(a|s)\right]$$

Williams, Ronald J. 1992. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning." *Machine Learning* 8 (3-4): 229–56.

Making sense of policy gradient

What does it do?

$$\nabla J(\theta) = \mathbb{E}_{s,a}\left[Q^{\pi}(s,a)\nabla_{\theta}\log \pi_{\theta}(a|s)\right]$$

- Gradient for each action is “weighted” based on the goodness

Efficiency of likelihood ratio

$$\nabla J(\theta) = \mathbb{E}_{s,a}\left[Q^{\pi}(s,a)\nabla_{\theta}\log\pi_{\theta}(a|s)\right]$$

- Gradient is “relative”

- To get a useful gradient, you need “many” samples
- Gradient is indirect

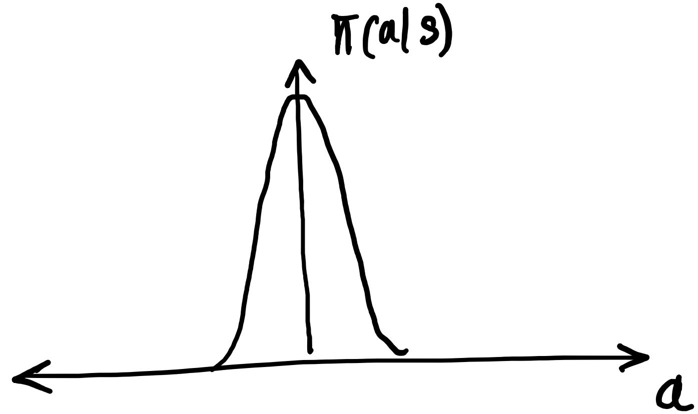
Policy collapse

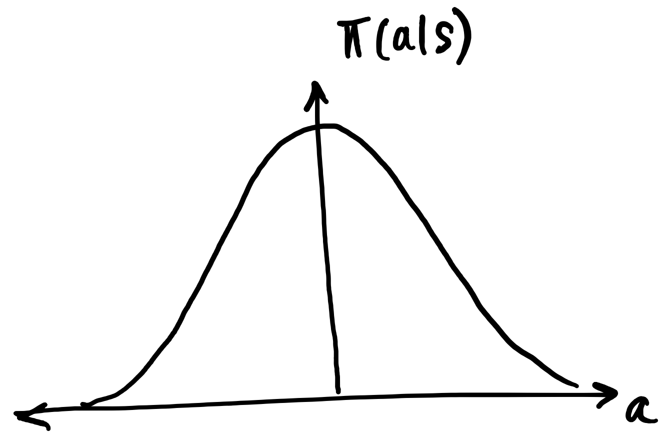
- Stochastic policy

- Policy often collapses into deterministic policy

- After which, no exploration

Exploration of policy gradient





Entropy

$$H(p) = -\sum_x p(x) \log p(x)$$

- We want to discourage policy collapse

- Revised objective

$$J(\theta) = \mathbb{E}_{s,a}\left[Q^\pi(s,a) \log \pi_\theta(a|s) + \beta H(\pi_\theta(s))\right]$$

Policy gradient pseudocode

for until satisfied **do**

  collect episode $(s_0, a_0, r_1, \dots)$ using π

  $J = \sum_i \gamma^i \left(g_i \log \pi_\theta(a_i|s_i) + \beta H(\pi_\theta(s_i))\right)$

  $\theta \leftarrow \theta + \alpha \nabla J$

end for

g_i

$H(\cdot)$

Policy gradient is on-policy

$$\nabla J(\theta) = \mathbb{E}_{s,a}\left[Q^{\pi}(s,a)\nabla \log \pi_{\theta}(a|s)\right]$$

$$\nabla J(\theta) = \sum_{s} d^{\pi}(s) \sum_{a} \pi(a|s)\left[Q^{\pi}(s,a)\nabla \log \pi_{\theta}(a|s)\right]$$

Proof of policy gradient

$V^{\pi}(s)$

$$J(\theta) = \sum_{s} P_{s_0}(s) V^{\pi}(s)$$

Policy gradient extensions

Variance of policy gradient

- What is the variance?

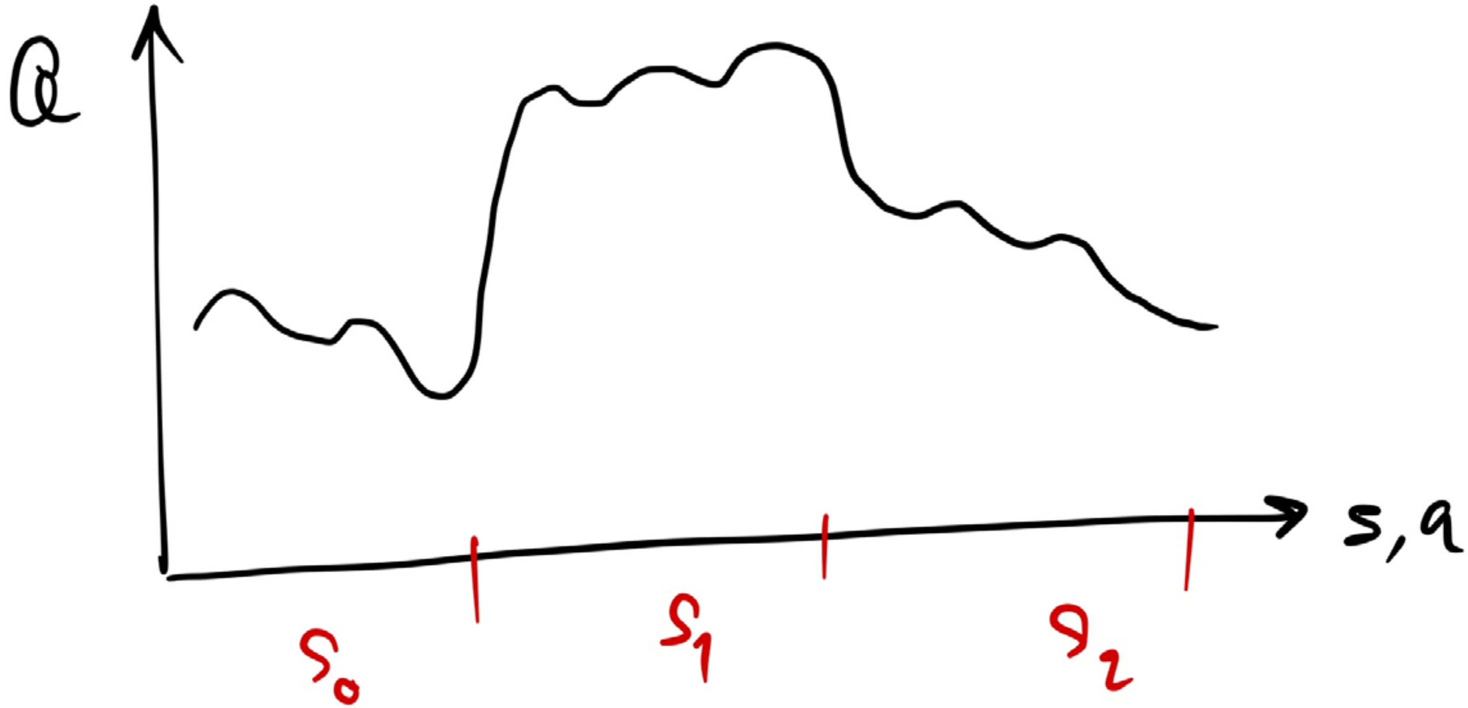
$$\mathrm{Var}(Q) = \mathbb{E}_{s,a}\left[Q(s,a) - \overline{Q}\right]^2$$

- Policy gradient has high variance

$$\nabla J(\theta) = \mathbb{E}_{s,a}\left[Q^{\pi}(s,a)\nabla \log \pi_{\theta}(a|s)\right]$$

- High variance slows down learning!

Variance in picture



Variance reduction

$$f(s,a) = Q(s,a) - b(s)$$

- If b(s) correlates well with Q(s,a)
- f(s,a) could have “lower” variance
- b(s) is called a “baseline”

- **There is no change in gradient! Why?**

$$\nabla J(\theta) = \mathbb{E}_{s,a}\left[(Q^{\pi}(s,a) - b(s))\nabla \log \pi_{\theta}(a|s)\right]$$

Baseline

$$\nabla J(\theta) = \mathbb{E}_{s,a}\left[(Q^{\pi}(s,a) - b(s))\nabla \log \pi_\theta(a|s)\right]$$

Baseline

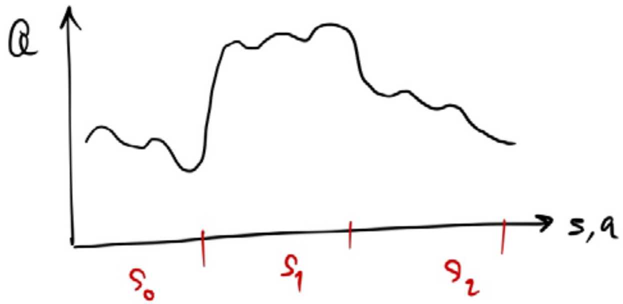
- What if b(s) becomes b(s, a)?
- Can we use the same argument?

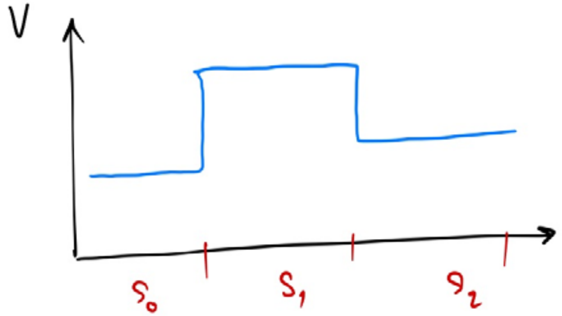
- What is a good baseline for Q(s,a)?

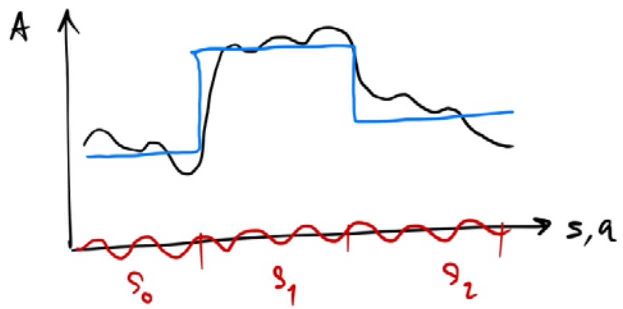
Advantage function

$$A(s,a) = Q(s,a) - V(s)$$

Baseline in action







Advantage Actor-critic (A2C)

$$\nabla J(\theta) = \mathbb{E}_{s,a}\left[A(s,a)\nabla \log \pi_\theta(a|s)\right]$$

$$A(s,a) = Q(s,a) - V(s)$$

- **Advantage = Critic**
- **Policy = Actor**
- Need to model either A or Q or V
- How to do it efficiently?

Sutton, Richard S., David Mcallester, Satinder Singh, and Yishay Mansour. 1999. "Policy Gradient Methods for Reinforcement Learning with Function Approximation." *In Advances in Neural Information Processing Systems 12*, 1057–63.

Practical considerations

Parallel environments

N-step exploration

A slew of returns

1. $\sum_{t=0}^{\infty} r_t$: total reward of the trajectory.
2. $\sum_{t'=t}^{\infty} r_{t'}$: reward following action a_t.
3. $\sum_{t'=t}^{\infty} r_{t'} - b(s_t)$: baselined version of previous formula.
4. $Q^{\pi}(s_t, a_t)$: state-action value function.
5. $A^{\pi}(s_t, a_t)$: advantage function.
6. $r_t + V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$: TD residual.

N-step TD Residual

$$A_t \approx r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^n V(s_{t+n}) - V(s_t)$$

Mnih, Volodymyr, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. "Asynchronous Methods for Deep Reinforcement Learning." *arXiv [cs.LG]*. arXiv. http://arxiv.org/abs/1602.01783.

Schulman, John, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. "High-Dimensional Continuous Control Using Generalized Advantage Estimation." *arXiv Preprint arXiv:1506.02438*, 1–14.

Generalized Advantage

- Lambda-weighted infinite sum of many-step advantage functions
- Lambda for advantage functions

$$A_t^{(\gamma,\lambda)} = \sum_{i=0}^{\infty} (\gamma\lambda)^i \delta_{t+i}$$

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

- We only sum to “n” (n >= 5)

Schulman, John, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. “High-Dimensional Continuous Control Using Generalized Advantage Estimation.” *arXiv Preprint arXiv:1506.02438*, 1–14.

A2C pseudocode

for until satisfied **do**

  collect n step $(s_0, a_0, r_1, \ldots, s_{n-1}, a_{n-1}, r_n, s_n)$ using π

  $q_i = \sum_{j=0}^{n} \gamma^j r_{i+j+1} + \gamma^n V_\phi(s_n)$

  $\nabla J(\theta) = \sum_{i=0}^{n} \gamma^i \left[q_i - V_\phi(s_i)\right] \nabla \log \pi_\theta(a_i|s_i)$

  $\nabla L(\phi) = \sum_{i=0}^{n} \left[V_\phi(s_i) - q_i\right] \nabla V_\phi(s_i)$

  $\theta \leftarrow \theta + \alpha_1 \nabla J$

  $\phi \leftarrow \phi + \alpha_2 \nabla L$

end for

Stability of critics

- **Semi-gradient could diverge**
- One-step return might be unstable
- You might need to use target network in such cases
- Usually N > 5, you don’t need (but using might give you even more stable training)

Sample efficiency

- Not able to reuse data
- Efficiency is not great
- Off-policy is much needed
- **Off-policy actor and critic**

Off-policy actor-critic

Off-policy critic

$$\nabla J(\theta) = \mathbb{E}_{s,a}\left[Q_\phi(s,a)\nabla \log \pi_\theta(a|s)\right]$$

- If Q learns from TD (one or many steps)

- How to learn Q off-policy?

- If one step, we could use expected SARSA

Off-policy policy gradient

- Objective function (proposed by the paper)

$$J(\theta) = \sum_{s} d^b(s) V^\pi(s)$$

- It is not the same as on-policy one!

$$J(\theta) = \sum_{s} P_{s_0}(s) V^\pi(s)$$

Degris, Thomas, Martha White, and Richard S. Sutton. 2012. “Off-Policy Actor-Critic.” *arXiv [cs.LG]*. arXiv. https://doi.org/10.1.1.385.4471.

Motivation

$$J(\theta) = \sum_s d^b(s) V^\pi(s)$$

- We have the distribution already

- Reweighting is not that bad

Off-policy policy gradient

$$\nabla_\theta J(\theta) \approx \sum_s d^b(s) \sum_a Q^\pi(s,a) \nabla_\theta \pi_\theta(a|s)$$

- Why approximation?

Proof of off-policy gradient

Approximation vs Oracle

$$\nabla_\theta J(\theta) = \sum_s d^b(s) \left[\sum_a Q^\pi(s,a) \nabla \pi(a|s) + \pi(a|s) \nabla Q^\pi(s,a) \right]$$

- **Oracle:**
  - Start from S
  - Update for S
  - Unroll, continue on-policy
- **Approximation:**
  - Start from S
  - Update for S

Off-policy gradient

- It gives a “good enough” gradient
- It guarantees to improve the policy
- But the fixed point for policy might not be as good (a local minima is not as good)

Deterministic policy gradient (DPG)

Motivation

- Deterministic policy simplifies much of the policy gradient
- **Gradient has very low variance!**
- Off-policy is trivial
- Potential efficiency gain

Deterministic Policy Gradient

- Deterministic policy $\pi_\theta(s) = a$

- Objective function

$$J(\theta) = \sum_s P_{s_0}(s) V^\pi(s) = \sum_s P_{s_0}(s) Q^\pi(s,a)|_{a=\pi(s)}$$

- Deterministic policy gradient

$$J(\theta) = \sum_s d^\pi(s) \nabla_a Q^\pi(s,a)|_{a=\pi(s)} \nabla_\theta \pi_\theta(s)$$

Policy improvement = backprop on critic

$$\nabla_\theta J(\theta) = \sum_s d^\pi(s) \nabla_a Q_\phi(s,a)|_{a=\pi(s)} \nabla_\theta \pi_\theta(s)$$

- We use a neural net as Q (critic)
- Critic "tells" what is a better action
- Very low variance
  - Single sample can make progress
- Easy overfit, critic needs to be "ahead"

Needs exploration

- Policy during exploration

$$a = \pi_\theta(s) + \epsilon$$

Connection with DQN

- DQN = Explicit max over discrete actions

$$\pi(s) = \operatorname*{argmax}_a Q_\phi(s, a)$$

- DPG = Climb to the local max action

$$J(\theta) = \sum_s d^\pi(s) \nabla_a Q_\phi(s, a)|_{a=\pi(s)} \nabla_\theta \pi_\theta(s)$$

Off-policy DPG

- Objective function

$$J(\theta) = \sum_s d^b(s) Q^\pi(s,a)|_{a=\pi(s)}$$

- Gradient

$$J(\theta) \approx \sum_s d^b(s) \nabla_a Q^\pi(s,a)|_{a=\pi(s)} \nabla_\theta \pi_\theta(s)$$

- Same argument with off-policy gradient

Off-policy critic

- Deterministic policy could evaluate from off-policy using SARSA-like
- (s, a, r, s')

$$\delta = \left(r + \gamma Q_\phi(s', \pi(s'))\right) - Q_\phi(s, a)$$

$$L(\phi) = \frac{1}{2}\delta^2$$

$$\nabla_\phi L(\phi) = -\delta \nabla_\phi Q_\phi(s, a)$$

Proof of DPG

$$J(\theta) = \sum_s d^\pi(s) \nabla_a Q^\pi(s,a)|_{a=\pi(s)} \nabla_\theta \pi_\theta(s)$$

Deterministic policy gradient extensions

Deep Deterministic Policy Gradient (DDPG)

- DPG uses one-step bootstrapping which is unstable
- DDPG introduces:
  - Target network both actor and critic
  - Replay
- DDPG = DQN for continuous control

Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. “Continuous Control with Deep Reinforcement Learning.” *arXiv [cs.LG]*. arXiv. https://doi.org/10.1561/2200000006.

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' 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:

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

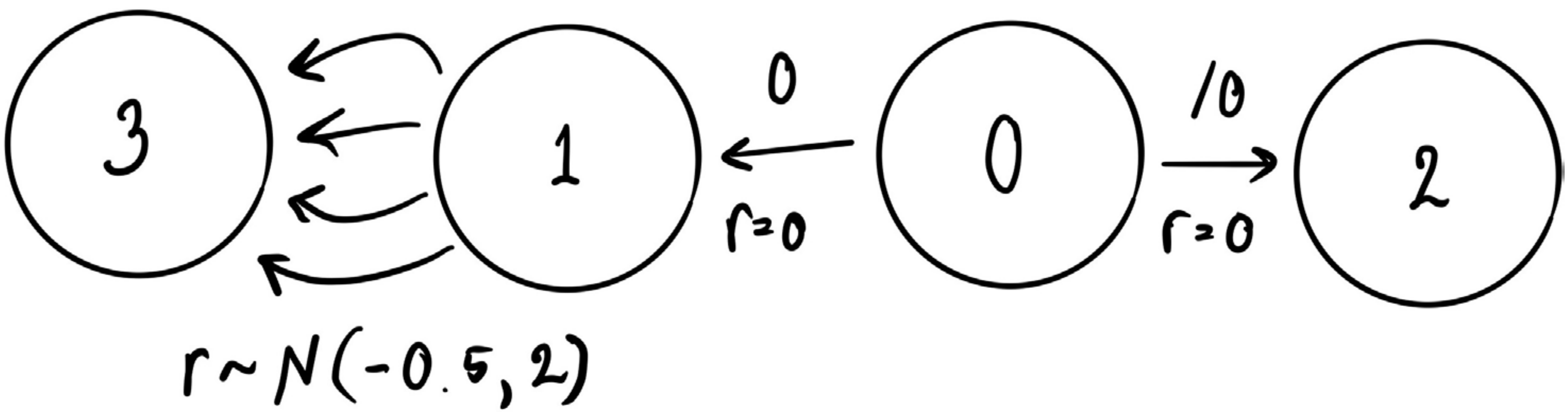
    Update the target networks:

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

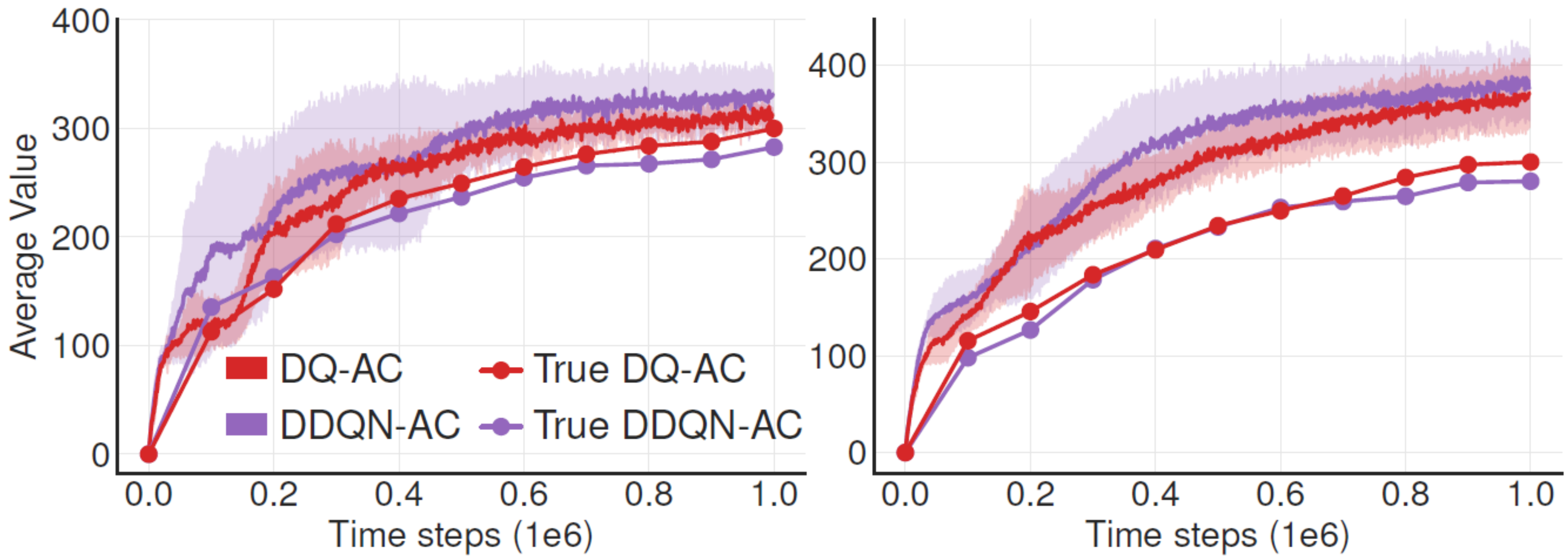
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Maximization bias



Maximization bias in DDPG



(a) Hopper-v1 (b) Walker2d-v1

- Double DQN doesn't work as well
- Because the policy slowly changes, the value functions are not "independent" enough

Fujimoto, Scott, Herke van Hoof, and David Meger. 2018. "Addressing Function Approximation Error in Actor-Critic Methods." *arXiv [cs.AI]*. arXiv. https://doi.org/10.1089/lrb.2012.0022.

MISC.

Action dependent baseline

- PG has high variance because it uses “indirect gradient”

$$\nabla J(\theta) = \mathbb{E}_{s,a}\left[Q^{\pi}(s,a)\nabla_{\theta}\log\pi_{\theta}(a|s)\right]$$

- DPG has lower variance because it can “backprop”

$$\nabla J(\theta) = \sum_{s} d^{\pi}(s)\nabla_{a}Q_{\phi}(s,a)|_{a=\pi(s)}\nabla_{\theta}\pi_{\theta}(s)$$

Action dependent baseline

- Can we combine the two?
- Q-Prop

$$\nabla J(\theta) = \mathbb{E}_{s,a}\left[(Q^\pi(s,a) - \overline{Q}(s,a))\nabla_\theta \log \pi_\theta(a|s)\right] + \mathbb{E}_s\left[\nabla_a Q(s,a)|_{a=\pi(s)} \nabla_\theta \pi_\theta(s)\right]$$

- Taylor expansion (first order)

$$\overline{Q}(s,a) = Q(s,\pi(s)) + \nabla_a Q(s,\pi(s))(a - \pi(s))$$

Gu, Shixiang, Timothy Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. 2016. "Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic." *arXiv [cs.LG]*. arXiv. http://arxiv.org/abs/1611.02247.

Policy gradient from minimizing KL

- If we look at Q as "unnormalized" policy
  - A little bit sharper of Q is exp(Q)
  - This is our target policy
- We could use a KL:

$$\pi = \underset{\pi \in \Pi}{\operatorname{argmin}} \ \mathrm{D}_{KL}\left(\pi(\cdot|s) \middle\| \frac{\exp(Q(\cdot, s))}{Z}\right)$$

- Minimizing KL is an optimization task

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor." *arXiv [cs.LG]*. arXiv. http://arxiv.org/abs/1801.01290.

Policy gradient from minimizing KL

- KL policy gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}_s \left[\nabla_\theta \mathrm{D}_{KL} \left(\pi(\cdot|s) \middle\| \frac{\exp(Q(\cdot, s))}{Z} \right) \right]$$

- Z is a constant, ignored
- Policy improve to Q
- Policy eval: Q gets even sharper
- Repeat

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor.” *arXiv [cs.LG]*. arXiv. http://arxiv.org/abs/1801.01290.

Assignments

- Pull chula_rl (beware of conflicts; back up)
- A2C with continuous Cartpole
- DDPG with continuous Cartpole