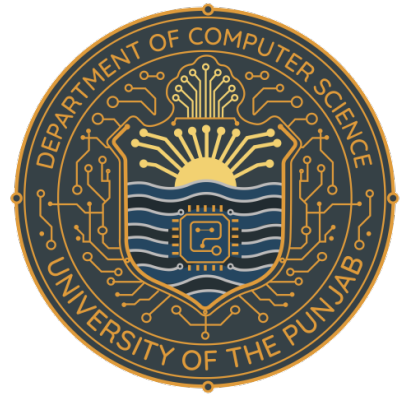
CS-566 Deep Reinforcement Learning

Improved Policy-Based Learning IV: Deteministic Policy Gradient



Nazar Khan
Department of Computer Science
University of the Punjab

Deterministic Policy Gradient[1] (DPG)

Actor-Critic recap:

- Actor-critic methods combine policy and value learning
- Can reduce variance and improve learning performance

Idea behind deterministic policy gradient:

- Learn a value function $Q_\phi(s, a)$
- Use it as a differentiable target to optimize the policy
- Policy *follows* the value function

[1] **silver2014deterministic**.

Deterministic Policy Gradient Objective

Collect data D and train a value network $Q_\phi(s, a)$.

Then optimize a deterministic policy $\pi_\theta(s)$ via:

$$J(\theta) = \mathbb{E}_{s \sim D} \left[\sum_{t=0}^{T} Q_\phi(s, \pi_\theta(s)) \right]$$

Chain rule gives:

$$\nabla_\theta J(\theta) = \sum_{t=0}^{T} \nabla_a Q_\phi(s, a) \cdot \nabla_\theta \pi_\theta(s)$$

Interpretation:

- Train $Q_\phi(s, a)$ from experience
- Update policy in the direction of actions with higher value
- 'Let the policy follow the value network'

Motivation for DDPG

Goal: Extend value-based deep RL (like DQN) to **continuous action spaces**.
Challenge in continuous spaces:

$$a^\star(s) = \arg\max_a Q^\star(s,a) \quad \text{is hard!}$$

DDPG solution:

- Use the gradient of $Q(s,a)$ w.r.t. the action to approximate $\max_a Q(s,a)$
- Deterministic policy actor: $\pi_\theta(s)$
- Critic estimates $Q_\phi(s,a)$

Based on:

- Deterministic policy gradient[2]
- NFQCA[3]

[2] **silver2014deterministic**.
[3] **hafner2011reinforcement**.

DDPG Algorithm Pseudocode

Randomly initialize critic network $Q_\phi(s, a)$ and actor $\pi_\theta(s)$ with weights ϕ and θ.
Initialize target network Q' and π' with weights $\phi' \leftarrow \phi$, $\theta' \leftarrow \theta$
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**
    $a_t = \pi_\theta(s_t) + \mathcal{N}_t$ 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 $y_i = r_i + \gamma Q_{\phi'}(s_{i+1}, \pi_{\theta'}(s_{i+1}))$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q_\phi(s_i, a_i))^2$
    Update the actor policy using the sampled policy gradient:

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

DDPG Algorithm Pseudocode

Update the target networks:

$$\phi' \leftarrow \tau\phi + (1-\tau)\phi'$$

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta'$$

end for
end for

DDPG Algorithm Overview

DDPG characteristics:

- Actor-critic, model-free
- Continuous actions
- Off-policy learning
- Replay buffer (like DQN)
- Target networks for stability (like DQN)

DDPG showed strong performance on physics control tasks:

- CartPole, Gripper, Walker, Car driving
- Learns from pixels in some settings

Key idea: Use deterministic policy + learned Q gradient to update actor.

DDPG Resources

Implementations:

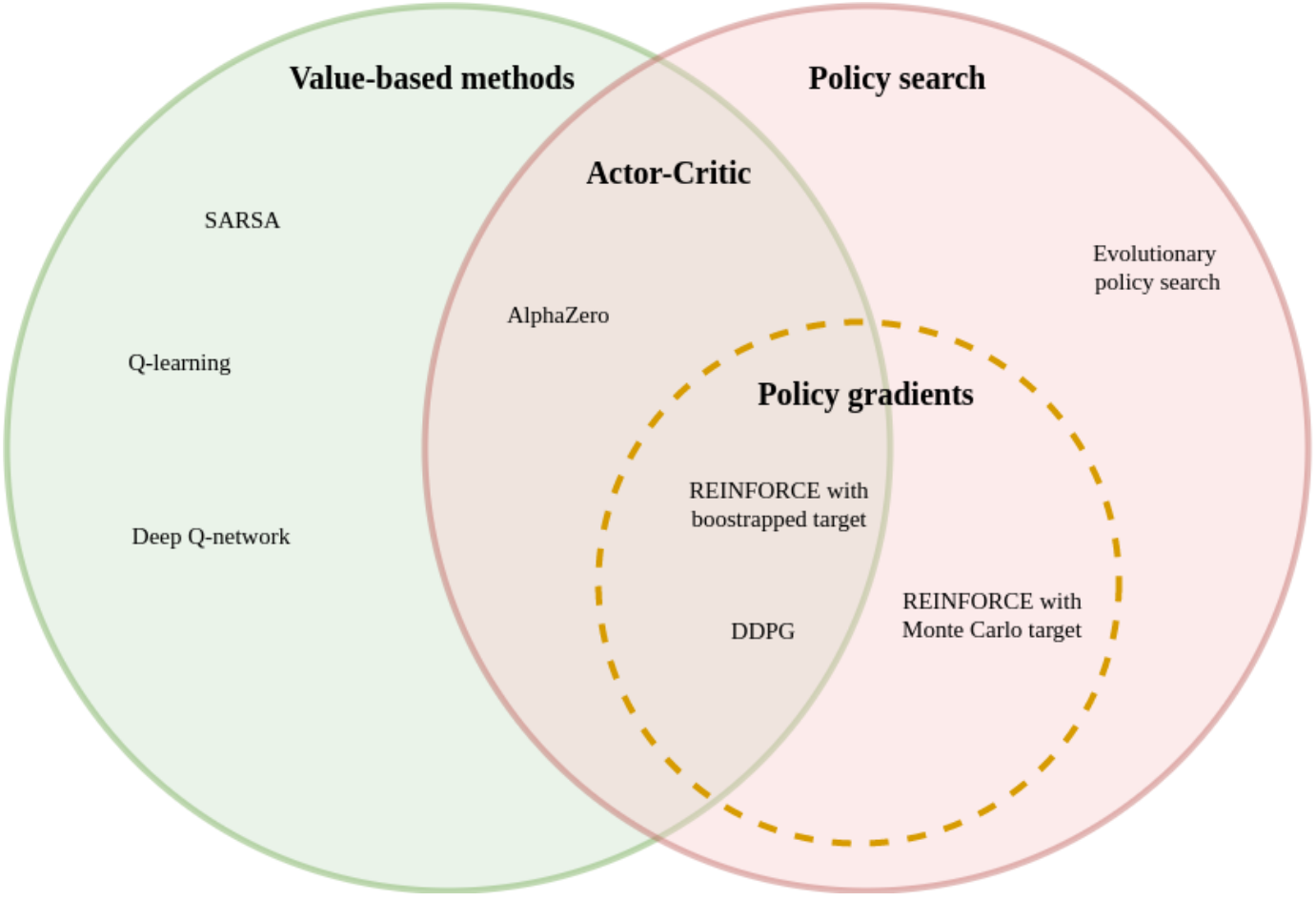
- Spinning Up (spinningup.openai.com)
- Stable-Baselines (stable-baselines.readthedocs.io)
- Original paper[4]

DDPG Summary:

- Bridges DQN ideas into continuous control
- Strong baseline in robotics and physics simulators
- Foundation for later methods (TD3, SAC)

[4] **lillicrap2015continuous**.

Value vs Policy vs Actor-Critic



Value-based, policy-based, and actor-critic methods[5].

[5] **moerland2021lecture**.