



**Iran University
of Science and
Technology**

In the Name of God

Reinforcement Learning in Control

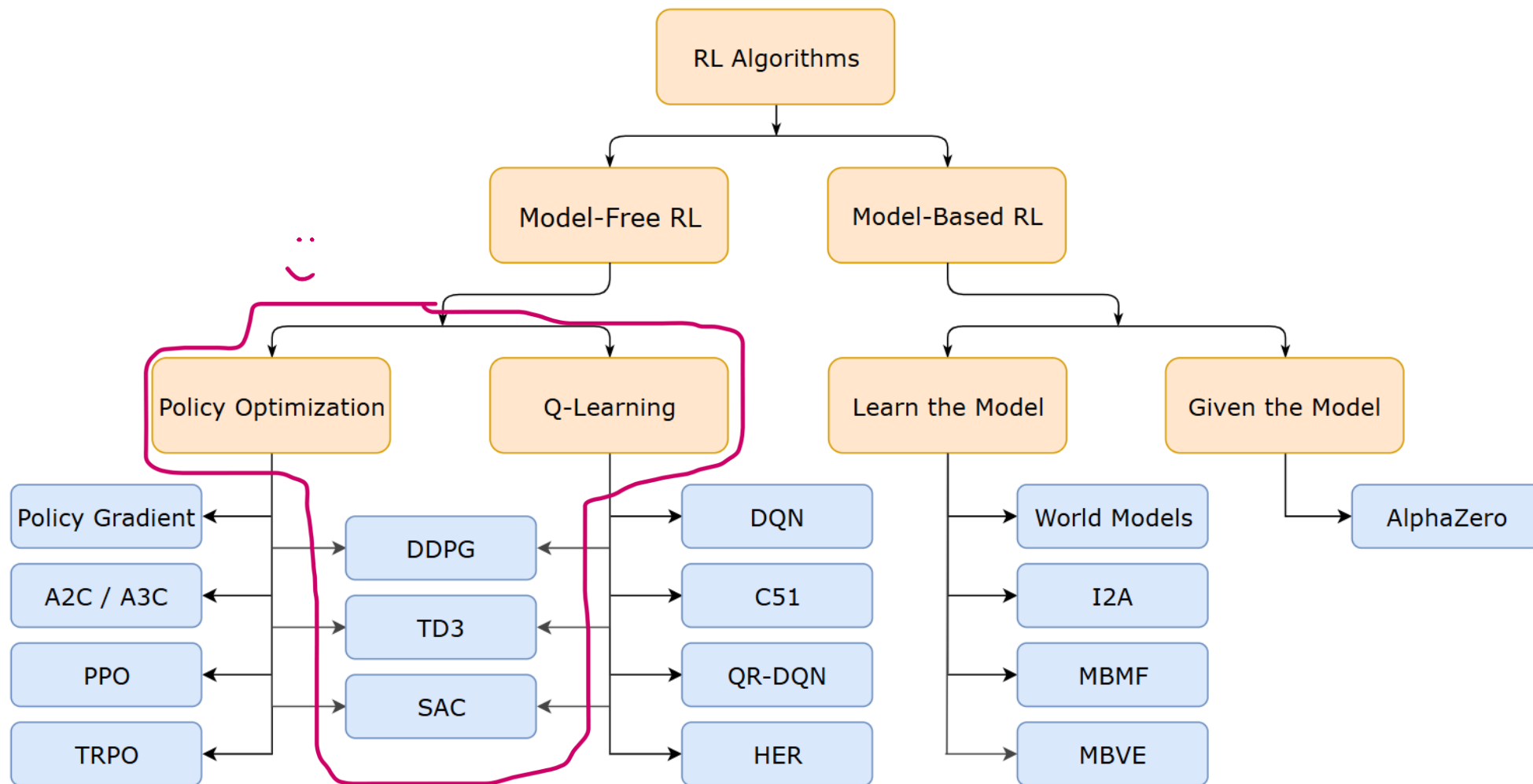
Dr. Saeed Shamaghdari

**Electrical Engineering Department
Control Group**

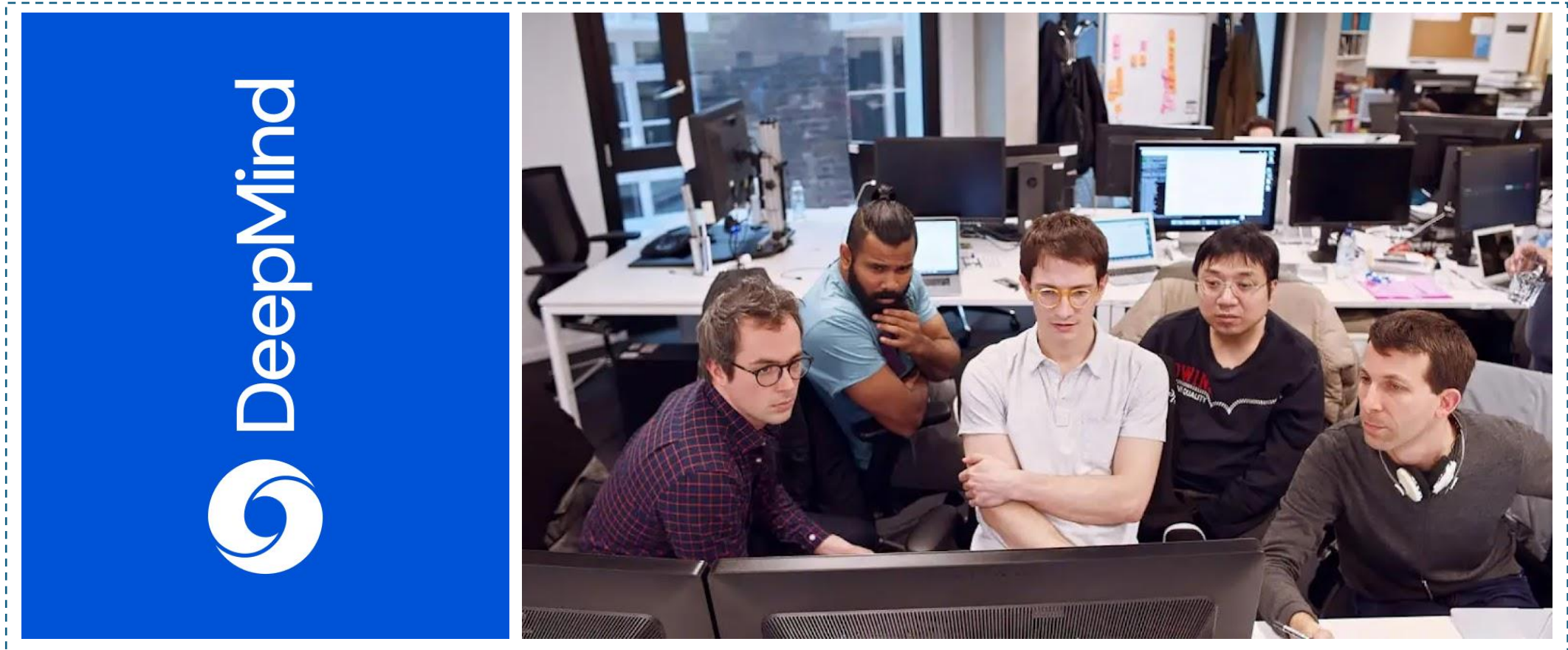
Fall 2025 | 4041

(Deep) Deterministic Policy Gradient

RL Algorithms



Once Again, Google DeepMind!



Deterministic Policy Gradient

Deterministic Policy Gradient Algorithms

David Silver

DeepMind Technologies, London, UK

Guy Lever

University College London, UK

Nicolas Heess, Thomas Degris, Daan Wierstra, Martin Riedmiller

DeepMind Technologies, London, UK

DAVID@DEEPMIND.COM

GUY.LEVER@UCL.AC.UK

*@DEEPMIND.COM

Deterministic Policy Gradient Algorithms ([Paper](#))

Deterministic Policy Gradient

“In this paper we consider deterministic policy gradient algorithms for reinforcement learning with continuous actions. The deterministic policy gradient has a particularly appealing form: it is the expected gradient of the action-value function. This simple form means that the deterministic policy gradient can be estimated much more efficiently than the usual stochastic policy gradient. To ensure adequate exploration, we introduce an off-policy actor-critic algorithm that learns a deterministic target policy from an exploratory behaviour policy. We demonstrate that deterministic policy gradient algorithms can significantly outperform their stochastic counterparts in high-dimensional action spaces.”

Deterministic Policy Gradient

Reminder Box

$$\pi_{\theta}(a|s) = \mathbb{P}[a|s; \theta]$$

→ Stochastic Policy

Sampling this stochastic policy and adjusting the policy parameters in the direction of greater cumulative reward.

Policy Gradients



Go Right

Deterministic Policy Gradient

“In this paper we instead consider *deterministic* policies”

$$a = \mu_{\theta}(s)$$

Q: Does a deterministic policy gradient actually exist?

“Deterministic policy gradient does indeed exist, and furthermore it has a simple model-free form that simply follows the gradient of the action-value function. deterministic policy gradient is the limiting case, as policy variance tends to zero, of the stochastic policy gradient.”



Stochastic PG vs Deterministic PG

*“From a practical viewpoint, there is a crucial difference between the stochastic and deterministic policy gradients. In the stochastic case, the policy gradient integrates over **both state and action spaces**, whereas in the deterministic case it only integrates over **the state space**. As a result, computing the stochastic policy gradient may require more samples, especially if the action space has many dimensions”*

*“In order to explore the full state and action space, a stochastic policy is often necessary. To ensure that our deterministic policy gradient algorithms continue to explore satisfactorily, we introduce an **off-policy learning algorithm**.”*

Stochastic PG vs Deterministic PG

“The basic idea is to choose actions according to a stochastic behaviour policy (to ensure adequate exploration), but to learn about a deterministic target policy (exploiting the efficiency of the deterministic policy gradient). We use the deterministic policy gradient to derive an off-policy actor critic algorithm that estimates the action-value function using a differentiable function approximator, and then updates the policy parameters in the direction of the approximate action-value gradient.”

Quick Review of RL

A Markov Decision Process with:
 An action Space \mathcal{A} and a state space \mathcal{S}
 Dynamics Distribution:

$$p(s_{t+1} | s_t, a_t)$$

$$p(s_{t+1} | s_1, a_1, \dots, s_t, a_t) = p(s_{t+1} | s_t, a_t)$$

A reward function $r: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

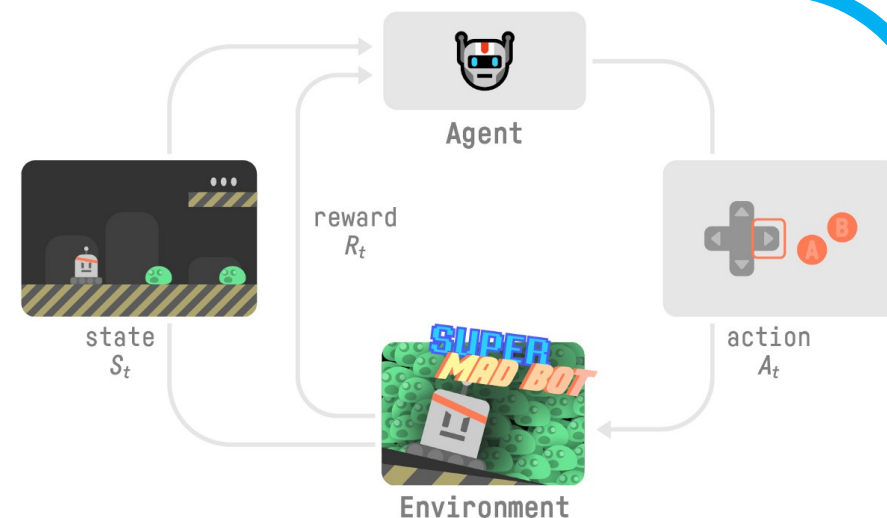
A policy $\pi_\theta : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$ is the set of probability measures on \mathcal{A}

$h_{1:T} = s_1, a_1, r_1, \dots, s_T, a_T, r_T$ over $\mathcal{S} \times \mathcal{A} \times \mathbb{R}$ (trajectory of states, actions and rewards)

$$r_t^\gamma = \sum_{k=t}^{\infty} \gamma^{k-t} r(s_k, a_k) \text{ (Total discounted reward or return)}$$

$$V^\pi(s) = \mathbb{E}[r_1^\gamma | S_1 = s; \pi]$$

$$Q^\pi(s, a) = \mathbb{E}[r_1^\gamma | S_1 = s, A_1 = a; \pi]$$



Quick Review of RL

Performance Objective:

$$J(\pi) = \mathbb{E} [r_1^\gamma | \pi]$$

(Improper) Discounted state distribution:

$$\rho^\pi(s') := \int_{\mathcal{S}} \sum_{t=1}^{\infty} \gamma^{t-1} \underbrace{p_1(s)}_{\text{Initial state distribution}} \underbrace{p(s \rightarrow s', t, \pi)}_{\text{(the density at state } s' \text{ after transitioning for } t \text{ time steps from state } s)} ds$$

Then we can write:

$$\begin{aligned} J(\pi_\theta) &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi_\theta(s, a) r(s, a) da ds \\ &= \mathbb{E}_{s \sim \rho^\pi, a \sim \pi_\theta} [r(s, a)] \end{aligned}$$

Quick Review of RL

Stochastic Policy Gradient:

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

Stochastic Actor-Critic:

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

Off-Policy Actor-Critic

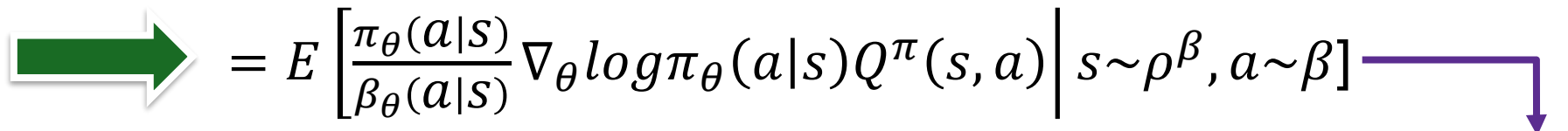
It is often useful to estimate the policy gradient *off-policy* from trajectories sampled from a **distinct behaviour** policy $\beta(a|s) \neq \pi_\theta(a|s)$

$$\begin{aligned} J_\beta(\pi_\theta) &= \int_{\mathcal{S}} \rho^\beta(s) V^\pi(s) ds \\ &= \int_{\mathcal{S}} \int_{\mathcal{A}} \rho^\beta(s) \pi_\theta(a|s) Q^\pi(s, a) da ds \end{aligned}$$

$$\begin{aligned} \nabla_\theta J_\beta(\pi_\theta) &\approx \int_{\mathcal{S}} \int_{\mathcal{A}} \rho^\beta(s) \nabla_\theta \pi_\theta(a|s) Q^\pi(s, a) da ds \\ &= \mathbb{E}_{s \sim \rho^\beta, a \sim \beta} \left[\frac{\pi_\theta(a|s)}{\beta_\theta(a|s)} \nabla_\theta \log \pi_\theta(a|s) Q^\pi(s, a) \right] \end{aligned}$$

Off-Policy Actor-Critic

$$\begin{aligned}
 \nabla_{\theta} J_{\beta}(\pi_{\theta}) &\approx \int_{\mathcal{S}} \int_{\mathcal{A}} \rho^{\beta}(s) \nabla_{\theta} \pi_{\theta}(a|s) Q^{\pi}(s, a) da ds \\
 &= E \left[\int_{\mathcal{A}} \nabla_{\theta} \pi_{\theta}(a|s) Q^{\pi}(s, a) \middle| s \sim \rho^{\beta} \right] \\
 &= E \left[\underbrace{\int_{\mathcal{A}} \beta_{\theta}(a|s)}_{E[\dots | a \sim \beta]} \underbrace{\frac{\pi_{\theta}(a|s)}{\beta_{\theta}(a|s)} \frac{\nabla_{\theta} \pi_{\theta}(a|s)}{\pi_{\theta}(a|s)}}_{\nabla_{\theta} \log \pi_{\theta}(a|s)} Q^{\pi}(s, a) \middle| s \sim \rho^{\beta} \right]
 \end{aligned}$$



$$\begin{aligned}
 &= E \left[\frac{\pi_{\theta}(a|s)}{\beta_{\theta}(a|s)} \nabla_{\theta} \log \pi_{\theta}(a|s) Q^{\pi}(s, a) \middle| s \sim \rho^{\beta}, a \sim \beta \right] \\
 &= \mathbb{E}_{s \sim \rho^{\beta}, a \sim \beta} \left[\frac{\pi_{\theta}(a|s)}{\beta_{\theta}(a|s)} \nabla_{\theta} \log \pi_{\theta}(a|s) Q^{\pi}(s, a) \right]
 \end{aligned}$$

Gradients of Deterministic Policies: Action-Value Gradients

Reminder Box

Policy Improvement:

$$\mu^{k+1}(s) = \operatorname{argmax}_a Q^{\mu^k}(s, a)$$


Q: What about continuous action space?

“In continuous action spaces, greedy policy improvement becomes problematic, requiring a global maximisation at every step. Instead, a simple and computationally attractive alternative is to move the policy in the direction of the gradient of Q , rather than globally maximising Q . ”

Gradients of Deterministic Policies: Action-Value Gradients

For each visited state s , the policy parameters θ^{k+1} , are updated in proportion to the gradient $\nabla_{\theta} Q^{\mu^k}(s, \mu_{\theta}(s))$.

Each state suggests a different direction of policy improvement; these may be averaged together by taking an expectation with respect to the state distribution $\rho^{\mu}(s)$.

$$\theta^{k+1} = \theta^k + \alpha \mathbb{E}_{s \sim \rho^{\mu^k}} \left[\nabla_{\theta} Q^{\mu^k}(s, \mu_{\theta}(s)) \right] \quad \text{By chain rule}$$


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

Deterministic Policy Gradient Theorem

A deterministic policy $\mu_\theta: \mathcal{S} \rightarrow \mathcal{A}, \theta \in \mathbb{R}^n$

Performance Objective $J(\mu_\theta) = \mathbb{E}[r_1^\gamma | \mu]$

Probability Distribution $p(s \rightarrow s', t, \mu)$

Discounted state distribution $\rho^\mu(s)$

Then:

$$\begin{aligned} J(\mu_\theta) &= \int_{\mathcal{S}} \rho^\mu(s) r(s, \mu_\theta(s)) ds \\ &= \mathbb{E}_{s \sim \rho^\mu} [r(s, \mu_\theta(s))] \end{aligned}$$

Deterministic Policy Gradient Theorem

Theorem 1 (Deterministic Policy Gradient Theorem)

Suppose that the MDP satisfies condition below:

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

These imply that $\nabla_\theta \mu_\theta(s)$ and $\nabla_a Q^\mu(s, a)$ exist *and that the deterministic policy gradient exists. Then*

$$\begin{aligned}\nabla_\theta J(\mu_\theta) &= \int_{\mathcal{S}} \rho^\mu(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a)|_{a=\mu_\theta(s)} ds \\ &= \mathbb{E}_{s \sim \rho^\mu} \left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a)|_{a=\mu_\theta(s)} \right]\end{aligned}$$

Limit of the Stochastic Policy Gradient

For a wide class of stochastic policies, including many bump functions, the deterministic policy gradient is indeed a special (limiting) case of the stochastic policy gradient.

Theorem 2

Consider a stochastic policy $\pi_{\mu_\theta, \sigma}$ where μ_θ is a deterministic policy and σ is a parameter controlling the variance. Then,

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

$$\begin{aligned} \sigma &\rightarrow 0 \\ \pi_{\mu_\theta, 0} &\equiv \mu_\theta \end{aligned}$$

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.

Deterministic Actor-Critic: On-Policy Q: Optimality?

Algorithm: On-Policy Actor-Critic

In this *deterministic actor-critic* algorithm, the critic uses Sarsa updates to estimate the action-value function.

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

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

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

Deterministic Actor-Critic: Off-Policy

Learn a deterministic target policy $\mu_\theta(s)$, from trajectories generated by an arbitrary stochastic behaviour policy $\pi(a|s) = \beta(a|s)$.

$$\begin{aligned} J_\beta(\mu_\theta) &= \int_{\mathcal{S}} \rho^\beta(s) V^\mu(s) ds \\ &= \int_{\mathcal{S}} \rho^\beta(s) Q^\mu(s, \mu_\theta(s)) ds \\ \nabla_\theta J_\beta(\mu_\theta) &\approx \int_{\mathcal{S}} \rho^\beta(s) \nabla_\theta \mu_\theta(a|s) Q^\mu(s, a) ds \\ &= \mathbb{E}_{s \sim \rho^\beta} \left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a) \Big|_{a=\mu_\theta(s)} \right] \end{aligned}$$

Deterministic Actor-Critic: Off-Policy **Q: Importance Sampling?**

“We now develop an actor-critic algorithm that updates the policy in the direction of the off-policy deterministic policy gradient.”

Algorithm: Off-Policy Actor-Critic

Substitute a differentiable action-value function $Q^w(s, a)$ in place of the true action-value function $Q^\mu(s, a)$. A critic estimates the action-value function $Q^w(s, a) \approx Q^\mu(s, a)$ off-policy from trajectories generated by $\beta(a|s)$, and uses Q-learning updates to estimate the action-value function.

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

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

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

For further details on the algorithms, including compatible function approximation and related methods, please refer to the [original paper](#). We will not go into more depth here.

Deep Deterministic Policy Gradient

Published as a conference paper at ICLR 2016

CONTINUOUS CONTROL WITH DEEP REINFORCEMENT LEARNING

**Timothy P. Lillicrap*, Jonathan J. Hunt*, Alexander Pritzel, Nicolas Heess,
Tom Erez, Yuval Tassa, David Silver & Daan Wierstra**

Google Deepmind

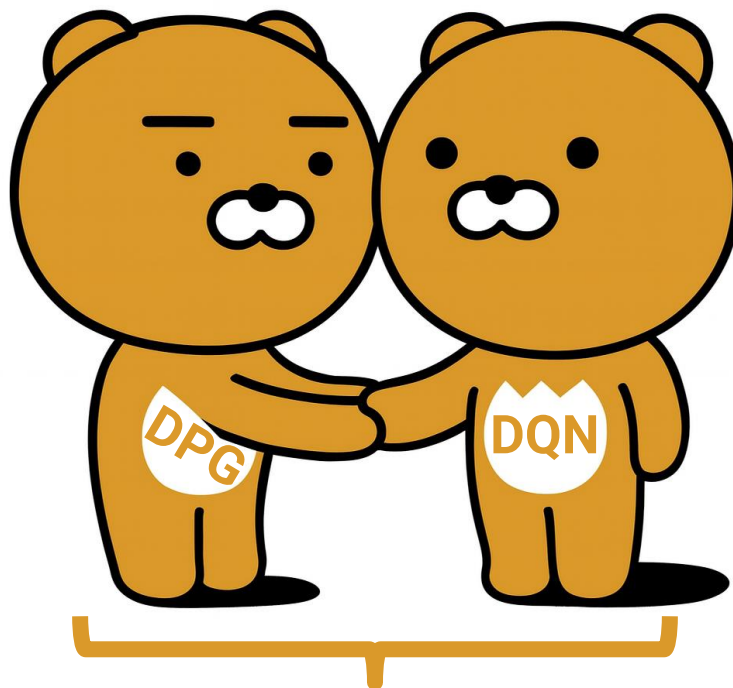
London, UK

{countzero, jjhunt, apritzel, heess,
etom, tassa, davidsilver, wierstra} @ google.com

Continuous Control with Deep Reinforcement Learning ([Paper](#))

Deep Deterministic Policy Gradient

learn policies “end-to-end”: directly from raw pixel inputs.



\approx ***DDPG***



Actor-Critic
Model-Free
Off-Policy
Based on DPG
For Continuous Action Space

DQN: Discussion Questions

Q: What is the main limitation of standard DQN?

It can only handle **discrete**
and **low-dimensional**
action spaces.

Q: Why can't DQN be directly applied to continuous action spaces?

It relies on finding the action that **maximizes** the action-value function, which in the continuous valued case requires an iterative optimization process at every step.

Well, simply discretize the action space! 🧐

Q: What problems arise when discretizing continuous action spaces, and why is it inefficient?

DQN: Discussion Questions

Q: What problems arise when discretizing continuous action spaces, and why is it inefficient?

A:

- Discretizing multi-dimensional actions causes exponential growth in the number of discrete actions (curse of dimensionality).
- Example: 7 DOF \times 3 values each $\rightarrow 3^7 = 2187$ actions.
- Fine-grained control requires even more actions.
- Naive discretization throws away continuity and structure, leading to redundancy and poor learning efficiency.
- Exploration becomes difficult, and training DQN-like models becomes nearly impossible.

DQN: Discussion Questions

Q: Is DQN only limited and problematic, or does it also have important advantages?

A:

- DQN uses a **replay buffer** to reduce sample correlations and stabilize learning.
- It uses a **target Q-network** to provide more consistent TD targets.
- As a result, DQN can **learn value functions stably and reliably** with deep networks.

DDPG

In the DPG paper, we saw

$$\begin{aligned}\nabla_{\theta} J_{\beta}(\mu_{\theta}) &\approx \int_{\mathcal{S}} \rho^{\beta}(s) \nabla_{\theta} \mu_{\theta}(a|s) Q^{\mu}(s, a) ds \\ &= \mathbb{E}_{s \sim \rho^{\beta}} \left[\nabla_{\theta} \mu_{\theta}(s) \nabla_a Q^{\mu}(s, a) |_{a=\mu_{\theta}(s)} \right]\end{aligned}$$

In the DDPG paper, the notation is slightly **different**:

$$\begin{aligned}\nabla_{\theta^{\mu}} J &\approx \mathbb{E}_{s_t \sim \rho^{\beta}} \left[\nabla_{\theta^{\mu}} Q(s, a | \theta^Q) |_{s=s_t, a=\mu(s_t | \theta^{\mu})} \right] \\ &= \mathbb{E}_{s_t \sim \rho^{\beta}} \left[\nabla_a Q(s, a | \theta^Q) |_{s=s_t, a=\mu(s_t)} \nabla_{\theta^{\mu}} \mu(s | \theta^{\mu}) |_{s=s_t} \right]\end{aligned}$$

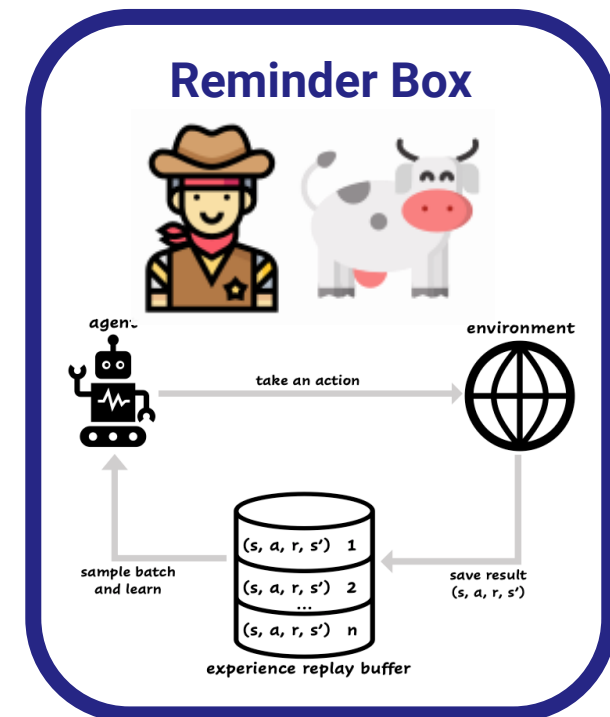
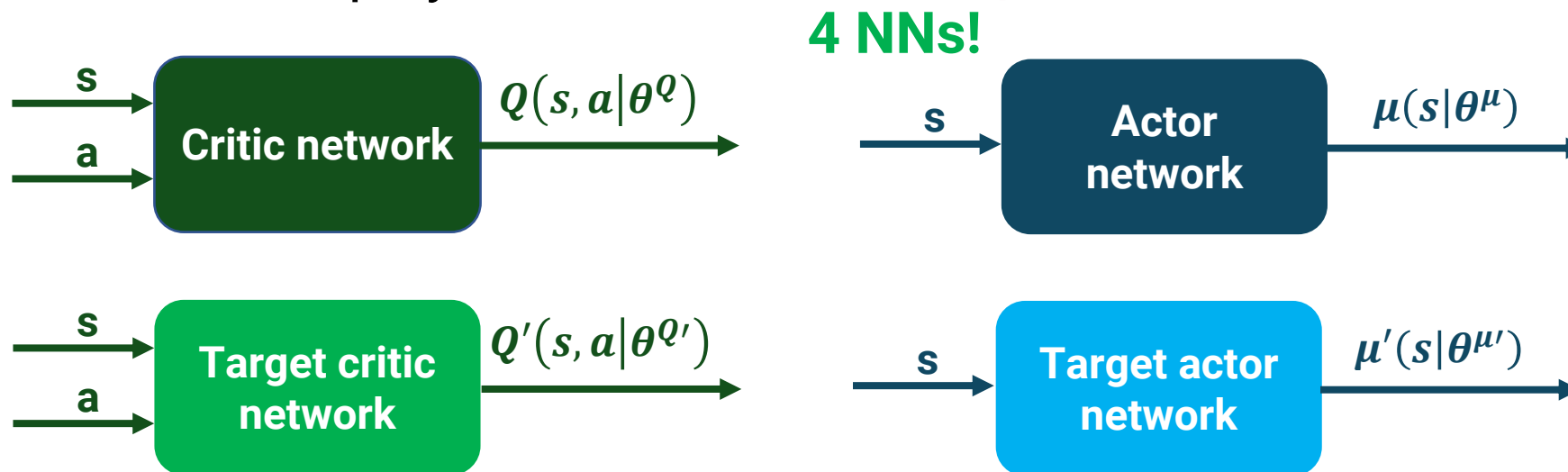
Reminder Box

$$\begin{aligned}R_t &= \sum_{i=t}^T \gamma^{(i-t)} r(s_i, a_i) \\ J &= \mathbb{E}_{r_i, s_i \sim E, a_i \sim \pi} [R_1]\end{aligned}$$

DDPG

Similar to DQN, we also use:

- separate target networks, **Q**: Why?
- and a replay buffer here



The weights of these target networks are then updated by having them slowly track the learned networks: $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$, $\tau \ll 1$

Batch Normalization

- **Manually** scaling features across environments is **difficult**.
- Use **Batch Normalization** to automatically normalize each feature (mean=0, variance=1).
- Keep running averages for use during testing/evaluation.
- Helps reduce covariance shift so each layer gets stable (“whitened”) inputs.
- Applied to:
 - state inputs
 - all layers of the μ -network
 - all layers of the Q -network before the action input
- Enables effective learning across tasks with different unit scales, without manual feature normalization.

Exploration Challenge

“A major challenge of learning in continuous action spaces is exploration. An advantage of off-policies algorithms such as DDPG is that we can treat the problem of exploration independently from the learning algorithm. We constructed an exploration policy μ' by adding noise sampled from a noise process \mathcal{N} to our actor policy

$$\mu'(s_t) = \mu(s_t | \theta_t^\mu) + \mathcal{N}$$

\mathcal{N} can be chosen to suit the environment.”

DDPG: Algorithm

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

End of Part I

RL

Control

