Reinforcement Learning workshop

Florian Goebels

October 3ed 2019

aggregate intellect

Agenda

- Day 1: RL Basics (theory)
- Day 2: Q-learning (coding)
- Day 3: Policy Gradient methods (coding)

Today’s Agenda

- Day 3: Policy Gradients
  - Background
  - Deep deterministic deep policy gradients

Q-learning

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    until S is terminal

TD $V(S)$ estimate

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right]$$

?

Better estimate of *V(S)*

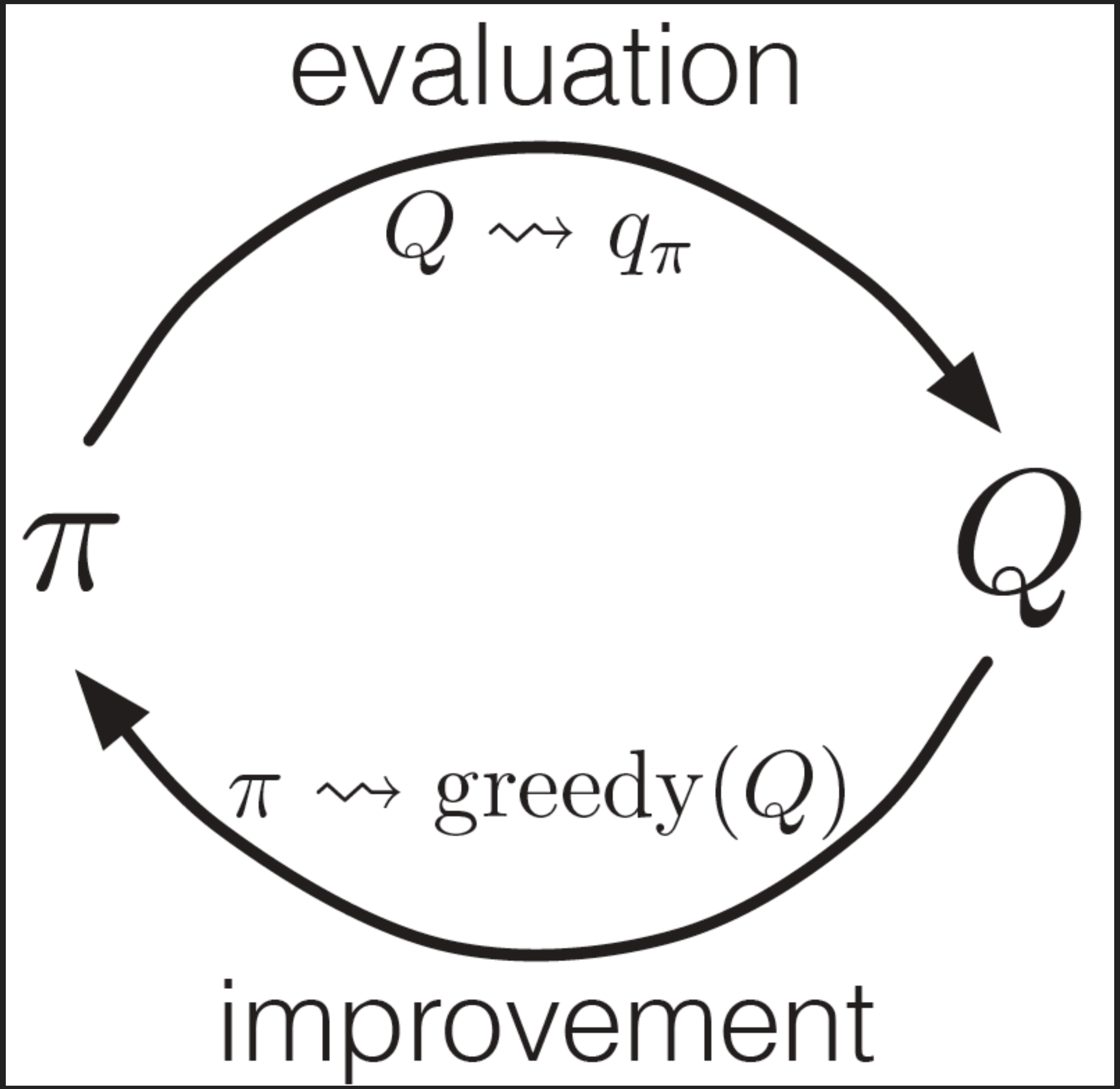
$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

TD ERROR

Current estimate of *V(S)*

Generalized Policy iteration

- Almost all RL algorithm are GPI
- Maintain both an approximate value function and an approximate policy
- Iteratively improve policy with respect to value function, and value function always drives to the value function of the current policy
- Overall process converges to to optimal policy and optimal value function



Deep Q-Learning

- In Q learning we iteratively update ***Q(A,S)*** via policy iteration
- Neural networks are general function approximators NN(S) -> V(A)
- Cost function:

Better estimate of *V(S)*

$$\left(\mathbf{NN}(S_t, a) - r + \gamma \max_a \mathbf{NN}(S_{t+1}, a)\right)^2$$

NN estimation of *V(S)*

Policy Gradient

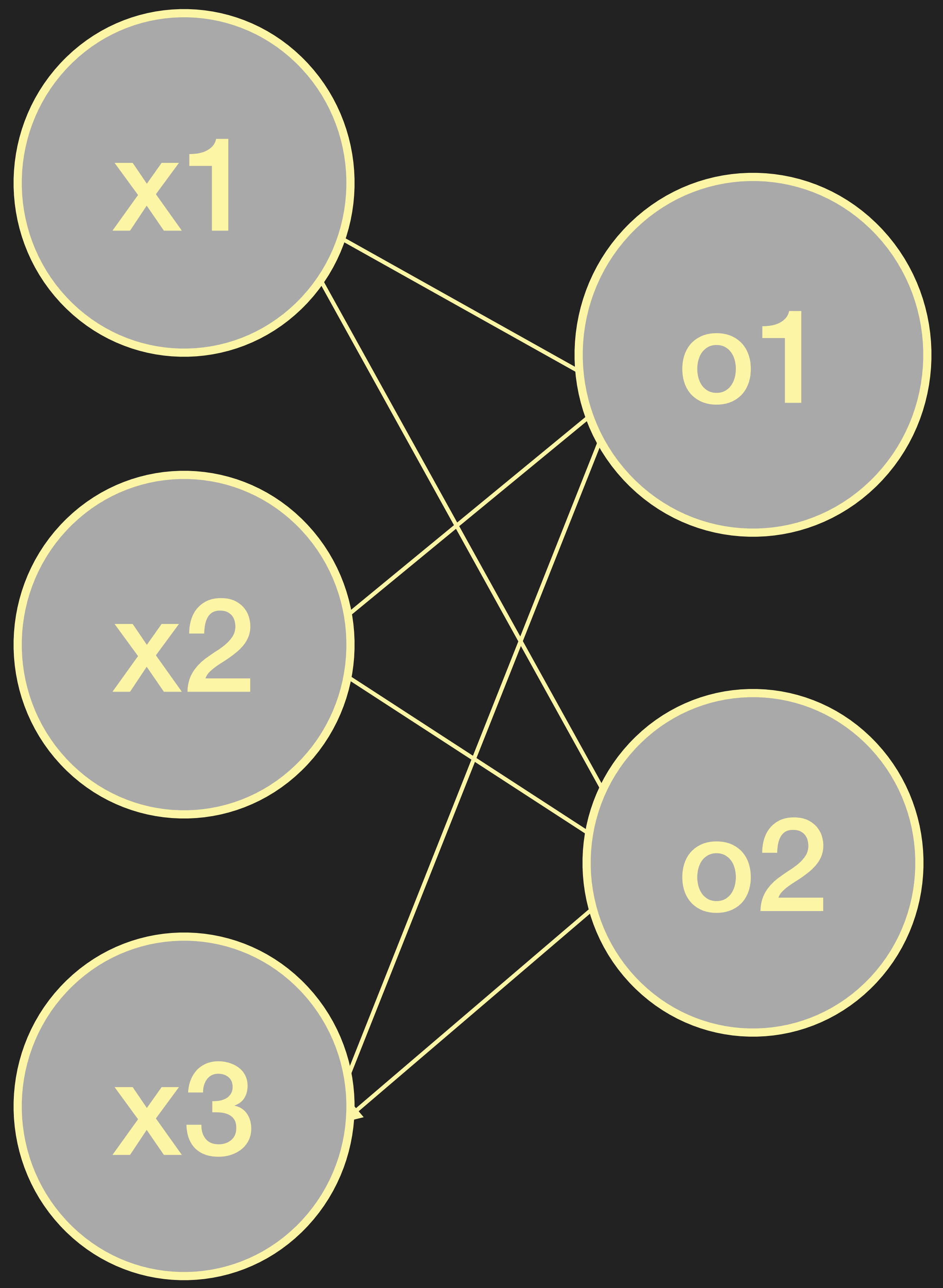
Weight matrix $\theta \in R^{d \times |A|}$

State features $x(s, a) \in R^{1 \times d}$

Hidden $h(s, a, \theta) = \theta \times x(s, a)$

Softmax activation $\pi(a \mid s, \theta) = \dfrac{e^{h(s,a,\theta)}}{\sum e^{h(s,a,\theta)}}$

Neural network propagation



Forward:

$$X * W = H$$

$$\sigma(H) = softmax(H) = Y$$

Backward:

$$Cost\ function: \quad C(y, y') = (y - y')^2$$

$$\frac{\partial C(y, y')}{\partial W} = X^T * (y' - y) * \sigma'(H)$$

Regression towards optimal policy

$$SW = \pi$$

$$SW = [0.2,0.8]$$

$$\pi * = [0.4,0.6]$$

$$C([0.2,0.8], [0.4,0.6]) = 0.28$$

We don’t know optimal policy

Policy gradient

Cost function $J(\theta)$

Gradient ascent $$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta)$$

Policy gradient $$\nabla_\theta J(\theta) = G \nabla ln\pi(a \mid s, \theta)$$

REINFORCE

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
    Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

Policy gradient

$$\nabla_{\theta} J(\theta) = G \nabla ln\pi(a \,|\, s, \theta) = \frac{\nabla \pi(a, s, \theta)}{\pi(a, s, \theta)}$$

$$\nabla \pi(a, s, \theta) = \frac{\partial \sigma(XW)}{\partial W} =$$

$$\frac{\partial \sigma(XW)}{\partial XW} * \frac{\partial XW}{W} = X^T * \sigma'(XW)$$

Deep deterministic policy gradient

Actor critic

Actor: Policy function controls our agent

$$\pi(A, A, \theta)$$

Critic: Value function measures how good actions are

$$q(s, a, w)$$

DDPG

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

DDPG

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:
  - Actor: input state - output policy
  - Critic: input state + policy - output q-value

DDPG

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

- Play game and use actor to create policy and store transitions in replay memory

DDPG

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: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$

- Sample mini batch
- predict policy from state with ***Q'*** and predict value of state with ***μ'***
- Calculate loss between y and model prediction of historic state, action pair

DDPG

Update the actor policy using the sampled policy gradient:

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

- Calculate policy gradient for actor

DDPG

Update the target networks:

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

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

- Update networks

License