Lecture 10:
(Deep) Deterministic Policy Gradient

B. Ravindran

PG for Continuous Action Spaces

- Policy Gradient Theorem (Discrete Actions):

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a \mid s)$$
$$= \mathbb{E}_{s\sim\mu^\pi(s), a\sim\pi_\theta}\left[q_\pi\left(S_t, A_t\right) \nabla \log \pi\left(A_t \mid S_t\right)\right]$$

- Policy Gradient Theorem (Continuous Actions):

$$\nabla J(\theta) = \int_{\mathcal{S}} \mu^\pi(s) \int_{\mathcal{A}} q_\pi(s,a) \nabla \pi(a \mid s) da\, ds$$
$$= \mathbb{E}_{s\sim\mu^\pi(s), a\sim\pi_\theta}\left[q_\pi\left(S_t, A_t\right) \nabla \log \pi\left(A_t \mid S_t\right)\right]$$

Continuous Action Spaces

- Hard to implement differentiable continuous controllers in many problems

- Expectation over both states and actions. If we can reduce the expectation to only over states, this might simplify gradient estimation

$$\nabla J(\theta) = \int_{\mathcal{S}} \mu^{\pi}(s) \int_{\mathcal{A}} q_{\pi}(s, a) \nabla \pi(a \mid s) da\, ds$$

Deterministic Policies

- Let the policy $\pi_\theta : S \rightarrow A$ be deterministic
- That is, $\pi_\theta(S)$ will output the action to be taken at state S

- The performance objective:

$$J(\theta) = \int_{\mathcal{S}} \mu^{\pi}(s) q_{\pi_\theta}(s, \pi_\theta(s)) ds$$

- Only one integral now as we have made the policies deterministic

Deterministic Policy Gradient

- What is the notion of gradient of deterministic policies?
- Continuous actions spaces allow us to think of change in actions w.r.t the policy parameter

$$\nabla_\theta J(\theta) = \int_{\mathcal{S}} \mu^\pi(s) \nabla_a q_\pi(s,a) \nabla_\theta \pi_\theta(s) \Big|_{a=\pi_\theta(s)} ds$$
$$= \mathbb{E}_{s\sim\mu^\pi}\left[\nabla_a q_\pi(s,a) \nabla_\theta \pi_\theta(s)|_{a=\pi_\theta(s)} \, ds \right]$$

- Expectation only over states
- Avoided max over actions by moving in the direction of the gradient of q_π

DPG Updates

- For a wide class of stochastic policies, the deterministic policy gradient is a limiting case of the stochastic policy gradient

- Update equations in actor critic framework:

$$\delta_t = r_t + \gamma \hat{q}\left(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}, \mathbf{w}\right) - \hat{q}\left(\mathbf{s}_t, \mathbf{a}_t, \mathbf{w}\right)$$

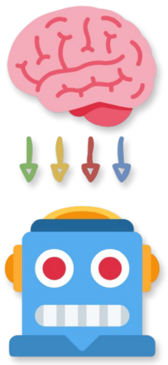
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha_{\mathbf{w}} \delta_t \nabla_{\mathbf{w}} \hat{q}\left(\mathbf{s}_t, \mathbf{a}_t, \mathbf{w}\right)$$

$$\theta \leftarrow \theta + \alpha_\theta \nabla_a \hat{q}\left(\mathbf{s}_t, \mathbf{a}_t, \mathbf{w}\right) \nabla_\theta \pi_\theta\left(\mathbf{s}_t\right)\big|_{a=\pi_\theta(s)}$$

DPG Exploration

- How do we ensure exploration?
  - If the environment is stochastic, then not an issue.
  - Otherwise use off-policy actor critic, where the behaviour policy differs from the estimation policy

- Requires compatible parametrizations



Deep Deterministic Policy Gradient

DDPG (Deep DPG)

- DDPG combines DPG with DQN
- The algorithm is off-policy. Behaviour policy:

$$\pi'(s) = \pi_\theta(s) + \underbrace{N}_{\text{Noise}}$$

- Uses replay buffer, improved sample efficiency
- Maintains separate target network parameters θ', w' and uses soft updates

- Uses Batch Normalization to normalize input state features and minimize covariate shift

DDPG Updates

- Critic Update

$$\delta_t = r_t + \gamma \hat{q}(\mathbf{s}_{t+1}, \pi_{\theta'}(\mathbf{s}_{t+1}), \mathbf{w}') - \hat{q}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha_{\mathbf{w}} \delta_t \nabla_{\mathbf{w}} \hat{q}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{w})$$

Target networks

- Actor Update

$$\theta \leftarrow \theta + \alpha_\theta \nabla_a \hat{q}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{w}) \nabla_\theta \pi_\theta(\mathbf{s}_t)|_{a=\pi_\theta(s)}$$

DDPG Algorithm

Randomly initialize critic network $\hat{q}(s, a, \mathbf{w})$ and actor $\pi_\theta(s)$ with weights $\mathbf{w}$ and θ
Initialize target network parameters $\mathbf{w}' \leftarrow \mathbf{w}$, $\theta' \leftarrow \theta$
Initialize replay buffer $\mathcal{R}$
for *episode=1, M* **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation s_1
  for *t=1,T* **do**
    Select action $a_t = \pi_\theta(s_t) + \mathcal{N}$ 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 $\mathcal{R}$

Collect samples

    Sample a random minimatch of N transitions (s_i, a_i, r_i, s_{i+1}) from $\mathcal{R}$
    Set $y_i = r_i + \gamma \hat{q}(s_{i+1}, \pi_{\theta'}(s_{i+1}), \mathbf{w}')$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - \hat{q}(s_i, a_i, \mathbf{w}))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_\theta J \approx \frac{1}{N}\sum_i \nabla_a \hat{q}(s_i, a_i, \mathbf{w}) \nabla_\theta \pi_\theta(s_i)|_{a=\pi_\theta(s)}$$

Update actor & critic

    Update the target parameters:

$$\mathbf{w}' \leftarrow \tau \mathbf{w} + (1-\tau)\mathbf{w}'$$

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

Update target network

  end
end