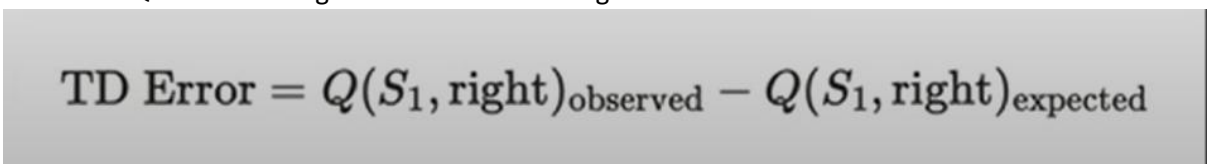
Assignment 1 – Task 1

1) Q-Learning

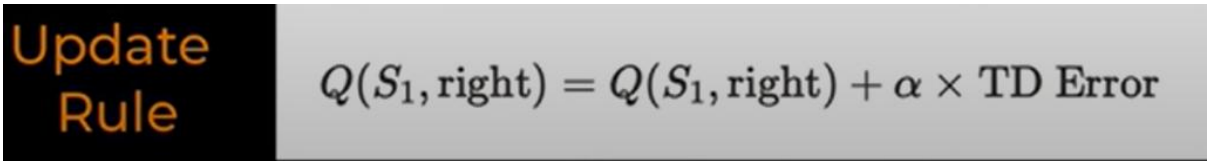
There are two types of methods namely Value-Based Methods and policy based methods. Value based methods find the maximum value of the reward and then find the optimal policy while policy based methods directly find the optimal policy. Q-learning is a value based method.

The possible actions are left, right, up and down.

The q-value for state S1 if right action is taken is the sum of reward of state S2 and the maximum Q value we can get from state S2. Here gamma is the discount factor.



Since we had a Q value earlier we will find a Temporal difference error ie the difference between the current Q value for the given state and action pair and that of the observed state and action pair.



This is the update rule for q value where alpha is the rate at which we are updating our q values.

Q learning is an off-policy algorithm. First we will randomize all the q values and then we take random initial actions till the q values stabilize and after that we find the optimal policy. The stabilization takes place by taking random actions initially and then continuing with the actions having the highest reward.

2) Monte-Carlo Control

Monte Carlo methods estimate value functions and improve policies based on complete episodes of interaction with the environment. They do not bootstraps, i.e., they wait until the end of an episode before updating estimates.

$$J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$$

Value Estimation (Prediction)

Let G_t be the **return** starting at time t:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}$$

We use G_t to update the value function:

For state-value function $V(s)$:

$$V(s) \leftarrow V(s) + \alpha(G_t - V(s))$$

For action-value function $Q(s,a)$:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(G_t - Q(s,a))$$

Where:

- α: learning rate.
- Updates only happen at the end of the episode (or after each first-visit to (s,a)).

3) Proximal Policy Optimization

$$\mathcal{L}^{\mathrm{CPI}}(\theta) = \hat{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)} \hat{A}_t \right] = \hat{E}_t \left[r_t(\theta) \hat{A}_t \right] \quad \text{(maximize)}$$

$$\mathcal{L}^{\mathrm{CLIP}}(\theta) = \hat{E}_t \left[\min \left(r_t(\theta) \hat{A}_t,\ \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right) \hat{A}_t \right) \right]$$

The first equation gives the function which is to be maximized ie the ratio of each action state pair of the current policy divided by the action state pair of the old policy multiplied by the advantage. But we want the policy to not change much after each iteration so we set a clip function which lies between 1+epsilon and 1-epsilon, and then take the minimum of the 2 ie either rt(theta) or the value of the clip function.

4) SAC

Objective Function

The goal in standard RL is to **maximize expected return**:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]$$

In **maximum entropy RL**, the objective becomes:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right) \right]$$

- $\mathcal{H}(\pi(\cdot|s)) = -\mathbb{E}_{a\sim\pi}[\log \pi(a|s)]$ is the entropy of the policy.
- α is the **temperature parameter** that determines the trade-off between exploration (entropy) and exploitation (reward).

1. Soft Q-Function

For a stochastic policy π(a|s), the soft Q-function is defined as:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}\left[\sum_{l=0}^{\infty} \gamma^l \left(r(s_{t+l}, a_{t+l}) + \alpha \mathcal{H}(\pi(\cdot|s_{t+l}))\right)\right]$$

The **soft Bellman backup** target is:

$$Q_{\text{target}} = r(s,a) + \gamma \mathbb{E}_{a' \sim \pi}\left[\min_{i=1,2} Q_{\theta_i'}(s', a') - \alpha \log \pi(a'|s')\right]$$

2. **Value Function**
Although SAC can use a value function V(s)V(s)V(s), most modern implementations **omit** the explicit value network and compute it from Q-values.

$$V(s) = \mathbb{E}_{a \sim \pi}\left[Q(s,a) - \alpha \log \pi(a|s)\right]$$

3. **Policy Update**
The policy is updated by minimizing the **expected KL divergence** between the policy and the exponentiated Q-function

$$J_{\pi} = \mathbb{E}_{s \sim D}\left[\mathbb{E}_{a \sim \pi}\left[\alpha \log \pi(a|s) - Q(s,a)\right]\right]$$

This encourages the policy to choose actions with high Q-values and high entropy.
4. **Critic Update**
Minimize the mean squared Bellman error for both Q-functions:

$$J_{Q_i} = \mathbb{E}_{(s,a,r,s') \sim D}\left[\left(Q_{\theta_i}(s,a) - Q_{\text{target}}\right)^2\right]$$

for i=1,2
5. **Target Networks**
The target networks are updated using **Polyak averaging**:

$$\theta_{\text{target}} \leftarrow \tau\theta + (1-\tau)\theta_{\text{target}}$$

where $\tau \in (0,1)$ is a small constant (e.g., 0.005).

5) Deep Deterministic Policy Gradient (DDPG)

DDPG is an **off-policy**, **model-free**, **actor-critic algorithm** for **continuous action spaces**. It combines ideas from **DQN** (Deep Q-Networks) and **Deterministic Policy Gradient**.

Objective

DDPG aims to **maximize expected return** by learning a deterministic policy $\mu(s)$:

$$J(\mu_\theta) = \mathbb{E}_{s\sim\mathcal{D}}\left[Q(s, \mu_\theta(s))\right]$$

Where:

- $Q(s, a)$: estimated action-value function.
- $\mathcal{D}$: replay buffer.

Core Components

1. **Critic (Q-function):**
   Trained by minimizing the Bellman error:

   $$L(\theta^Q) = \mathbb{E}_{(s,a,r,s')}\left[(Q(s, a) - y)^2\right]$$

   Where:

   $$y = r + \gamma Q_{\theta^{Q'}}(s', \mu_{\theta^{\mu'}}(s'))$$

   - $\theta^{Q'}$, $\theta^{\mu'}$: target networks (slow-moving copies).
2. **Actor (Policy):**
   Updated by applying the **Deterministic Policy Gradient Theorem**:

   $$\nabla_{\theta^\mu} J \approx \mathbb{E}_{s\sim\mathcal{D}}\left[\nabla_a Q(s, a)|_{a=\mu(s)} \cdot \nabla_{\theta^\mu}\mu(s)\right]$$

   This gradient pushes the policy toward actions that have higher Q-values.

Target Network Update

Use **Polyak averaging** to slowly update the target networks:

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

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

Where $\tau \ll 1$ (e.g., 0.005).

Exploration

Since the policy is **deterministic**, DDPG uses **noise** during training:

$$a = \mu(s) + \mathcal{N}_t$$

Where $\mathcal{N}_t$ is commonly **Ornstein-Uhlenbeck noise** or **Gaussian noise** to encourage exploration.