<u>Report 17/03</u>

<u>Tasks to do :</u>

1) Read about **Markov Decision Process**
2) Read about **Reinforcement Learning**
3) Think of how would we solve our network-friendly issue with **MDP.**

<u>What we did :</u>

4) Read about **Markov Decision Process**
   (from the youtube link (https://www.youtube.com/watch?v=i0o-ui1N35U) and the slides of David Silver)

We've seen that a Markov Decision Process is defined as :

- A set of stages s
- A set of actions
- A transition function T(s,a,s') (probability from a state s to be at a state s' with action a) (P(s'|s,a)).
- A reward function R(s,a,s')
- A discount value
- A start state

We also want to find the right **policy** (a function which gives an action for each state). The policy can either be deterministic or stochastic.

We also have a **utility** function (which is the sum of the discounted rewards). There is a discount value to decrease the impact of the reward toward the progression.

We also have an action-value function $Q_\pi(s,a)$ returning a value of that particular action.A q-state is a state with (s,a) before getting to s'.

We defined also a **value** function v(s) which gives the long-term value of state s.

The **Bellman equation** : (which gives the information of how good is to stay in that

$$\begin{aligned}
\mathcal{V}_\pi(s) &= \mathbb{E}[\mathcal{G}_t|\mathcal{S}_t = s] \\
&= \mathbb{E}[\mathcal{R}_{t+1} + \gamma\mathcal{R}_{t+2} + \gamma^2\mathcal{R}_{t+3} + \ldots|\mathcal{S}_t = s] \\
&= \mathbb{E}[\mathcal{R}_{t+1} + \gamma(\mathcal{R}_{t+2} + \gamma\mathcal{R}_{t+3} + \ldots)|\mathcal{S}_t = s] \\
&= \mathbb{E}[\mathcal{R}_{t+1} + \gamma\mathcal{G}_{t+1}|\mathcal{S}_t = s] \\
&= \mathbb{E}[\mathcal{R}_{t+1} + \gamma\mathcal{V}_\pi(s_{t+1})|\mathcal{S}_t = s]
\end{aligned}$$

state)

The idea behind this equation is to have an immediate reward and a discounted future value.

The goal is to maximize this value function.

We finally come with those two equations :

$V_*(s)=\max_{a\in A}(R_{as}+\gamma\sum_{s'\in S}P_{ass'}V_*(s')))$ (**Optimal state-value function**)
$Q_*(s,a)=R_{as}+\gamma\sum_{s'\in S}P_{ass'}\max_{a'\in A}Q_*(s',a')$ (**Optimal action-value function**)

5) Read about **Reinforcement Learning**

6) Think of how would we solve our network-friendly issue with **M**