L’apprentissage par renforcement (RL)
(Les bases, Q-Learning, Deep Q-Learning)

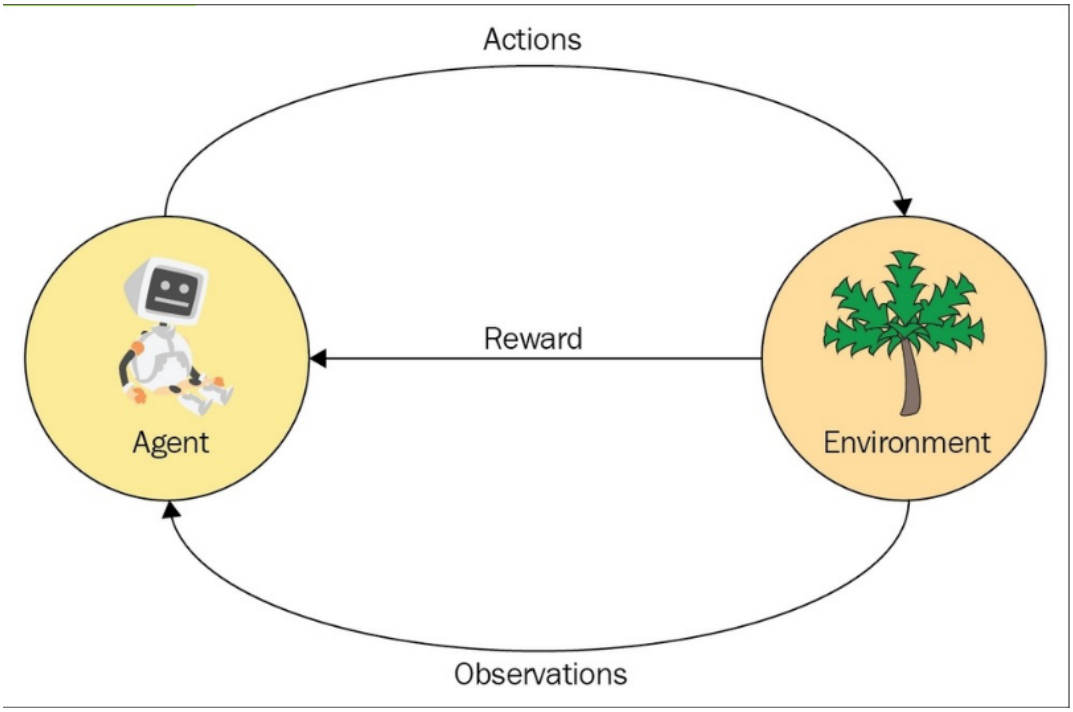
1 - L’apprentissage par renforcement (Reinforcement Learning) :

Dans l'apprentissage par renforcement, contrairement à l'apprentissage supervisé et non supervisé, vous n'avez pas besoin d'un ensemble de données pour apprendre, vous mettez simplement en place **un agent** dans **un environnement** qui explore et apprend ce qu'il faut faire, c'est-à-dire découvrir les bonnes et les mauvaises choses.

L’agent commence par **un état (state)** initial dans l’environnement et il a un ensemble **d’actions** à effectuer appelée **« Action Space »**. Quand l’agent exécute une action dans l’environnement, il va passer d’un état (state) à l’autre, et pour qu’il apprenne si l’action exécutée était la bonne, il reçoit une valeur de **récompense (reward)** de la part de l’environnement. La valeur de récompense aidera notre agent à apprendre en évitant les chemins qui apportent moins de récompense

En résumé, l’agent va apprendre le chemin des transitions des états qui va apporter le plus des récompenses.

La figure ci-dessous résume l’architecture de l’apprentissage par renforcement :



Les entités de l’apprentissage par renforcement :

Donc dans l’apprentissage par renforcement on a les entités suivantes :

- Agent.
- Récompense (Reward).
- Actions.
- Environnement.
- Etat (state) et aussi appelée observations.

Ci-dessous une définition résumée de chaque entité :

Agent : C'est l'acteur dans l'apprentissage par renforcement, c'est lui qui exécute les actions et apprend de ces derniers pour bien se comporter dans l'environnement et recevoir le maximum de récompenses.

Actions : Les actions sont les méthodes exécutées par l'agent dans un environnement et lui permettent de passer d'un état à l'autre. Chaque action effectuée par l'agent **donne lieu à une récompense** de la part de l'environnement. La décision de l'action à choisir est prise par la politique **(Policy)**.

Récompense (Reward) : une valeur numérique reçue de la part de l'environnement a l'agent par rapport à l'action exécutée dans un état, le but de l'agent est d'essayer d'avoir le maximum de ces récompenses.

Etat (State) : Chaque scénario que l'agent rencontre dans l'environnement est formellement appelé un état (state). L'agent passe d'un état à l'autre en effectuant des actions.

Environnement : C’est le monde qui contient l'agent et qui permet à l'agent d'observer l'état (state) de ce monde. Lorsque l'agent applique une action à l'environnement, ce dernier passe d'un état à l'autre, et envoi la valeur de récompense a l’agent.

Autres définitions dans le contexte :

Politique (π) : La politique, dénommée π (ou parfois $\pi(a|s)$), est une correspondance entre un certain état *S* et les probabilités de sélection de chaque action possible étant donné cet état.

Etat terminal (Terminal State) : Les états terminaux marquent la fin d'un épisode. Il n'y a pas d'états possibles après qu'un état terminal ait été atteint.

Episode : C'est de passer de l'état initial jusqu’à l'état finale, l'apprentissage se fait en plusieurs épisodes.

Le facteur de réduction (Discount Factor) : Le facteur de réduction (Discount Factor), généralement désigné par γ, est un facteur multipliant la récompense future attendue, et varie

dans l'intervalle [0,1]. Il contrôle l'importance des récompenses futures par rapport aux récompenses immédiates.

Remarque : Le facteur a le but de provoquer l'agent à terminer plus vite et donne l'importance aux récompenses immédiates et non pas au récompenses futures.

Exemple : Dans un jeu, le facteur de réduction a pour effet souhaitable de pousser les acteurs à essayer de gagner le plus tôt possible.

1 – 1 Exploration et Exploitation :

Exploration : L'agent doit essayer de nombreuses actions différentes dans de nombreux états différents afin d'essayer d'apprendre toutes les possibilités disponibles et de trouver le chemin qui maximisera sa récompense globale.

Exploitation : Utiliser les informations appris dans l'étape de l'exploration pour choisir la bonne action qui permet d'obtenir le plus de récompenses.

1 – 2 Value-function et Q-function :

Pour donner un peu d'intelligence à l'agent il existe une fonction qui représente la qualité de l'état dans lequel l'agent peut se trouver. La définition de cette fonction est ci-dessous.

Value function : la fonction Valeur est une mesure de la récompense globale attendue en supposant que l'agent se trouve dans l'état *S*.

Pour savoir dans quelle mesure une action donnée est bonne quand on est dans un état (state) précis on utilise la « Q-function ». La définition de cette fonction est ci-dessous :

Q-value (Q-function) : La fonction qui prédit le rendement attendu de l'exécution d'une action dans un état donné. Elle est également connue sous le nom de **« state-action value function »**.

Important : Ces deux fonctions seront utilisées dans les algorithmes d'apprentissage par renforcement. Chaque fonction à ces inconvénients et ses avantages, par exemple la « Q-function » est utile dans le « Q-Learning » et le « Deep Q-Learning » et la « value function » est en contrepartie utile dans « Actor-Critic (A2C) » (Lorsque l'espace d'action est grand).

1 – 3 Les algorithmes utilisés pour l'apprentissage par renforcement :

1.3.1 - Q-Learning : C'est un algorithme de l'apprentissage par renforcement, il commence d'abord par initialiser une table vide qui permet ensuite de stocker tous les « Q-values » de tous les pairs des états-actions possible (c'est-à-dire pour chaque état si on exécute une action quelle « Q-values » on obtient ?), Il met à jour cette table en utilisant **l'équation de Bellman**, tandis que le choix des actions dans cette algorithme est généralement effectué avec **une politique ε-Greedy**.

Il est très utile quand l'espace des états est petit.

La mise à jour de la « Q-value » est faite par l'équation suivante :

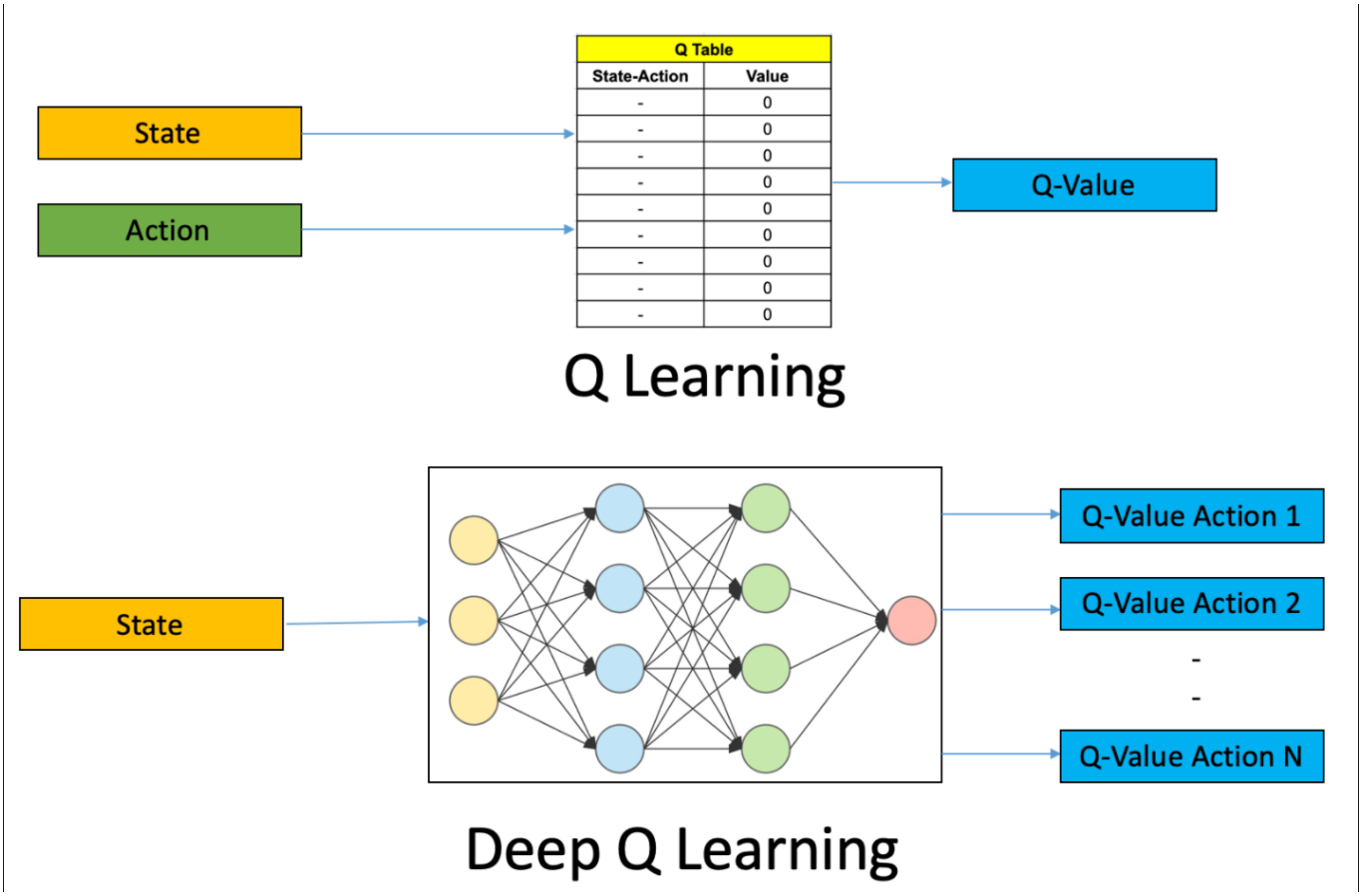
$$Q(s,a) = r(s,a) + \gamma \max_{a} Q(s',a)$$

Mais cet algorithme a un gros inconvénient. Lorsque l'espace d'état est grand, la taille du tableau « Q-values » peut devenir énorme, et la mémoire ne peut pas supporter cette taille, donc pour éliminer cet inconvénient et utiliser un grand espace d'état, nous utilisons l'algorithme de « **Deep Q-Learning** ».

1.3.2 - Deep Q-Learning : Les inconvénients de l'algorithme Q-Learning en ce qui concerne la taille de l'espace d'état ont incité les chercheurs à trouver de nouvelles solutions à ce problème. Les résultats de cette recherche ont abouti à l'algorithme « Deep Q-Learning ». Le mot « Deep » vient de « Deep Learning » basé sur **les réseaux de neurones**.

Dans cet algorithme on combine le principe des réseaux de neurones avec le principe du « Q-Learning ». Dans le réseau de neurone, les états de notre environnement seront l'entrée du modelé, et on obtient en sortie une « Q-value » pour chaque action, l'ensemble des « Q-value » seront utilisée après pour sélectionner une action lors de la phase de **l'exploitation**.

La figure ci-dessous schématise bien la différence entre le Q-Learning et le « Deep Q-Learning » tout en montrant les entrées et les sorties de notre réseau de neurone.



Le réseau de neurones a besoin de données pour apprendre et prédire les résultats, et puisque l'apprentissage par renforcement ne dispose pas d'un ensemble de données, nous obtiendrons nos propres données de l'environnement, les informations environnementales que nous utiliserons comme données d'apprentissage sont : état **(state)**, état suivant **(next_state)**, action, récompense **(reward)** et une valeur booléenne **(done)** qui indique si nous avons atteint notre état final.

Donc on un ensemble d'informations qui est obtenu à chaque transition, et cet ensemble est appelé une **expérience**. L'expérience est ensuite stockée dans un tampon (buffer) appelé **« Replay Buffer »**. L'apprentissage commence lorsque le **« Replay Buffer »** atteint une certaine taille (pour avoir suffisamment de données pour l'apprentissage).

Pour le bon fonctionnement du réseau de neurone, les données doivent être indépendantes, pour ça on va utiliser une technique pour réduire la corrélation temporelle dans les données en prenant des échantillons aléatoire de données pour être utilisée pour l'apprentissage. Cette technique est connue sous le nom de **« Replay Experience »**.

L'apprentissage du réseau de neurone ici est un apprentissage supervisé, donc on a besoin d'un target (comme le « label ») et une valeur « y » générée de la part du réseau, ces valeurs sont

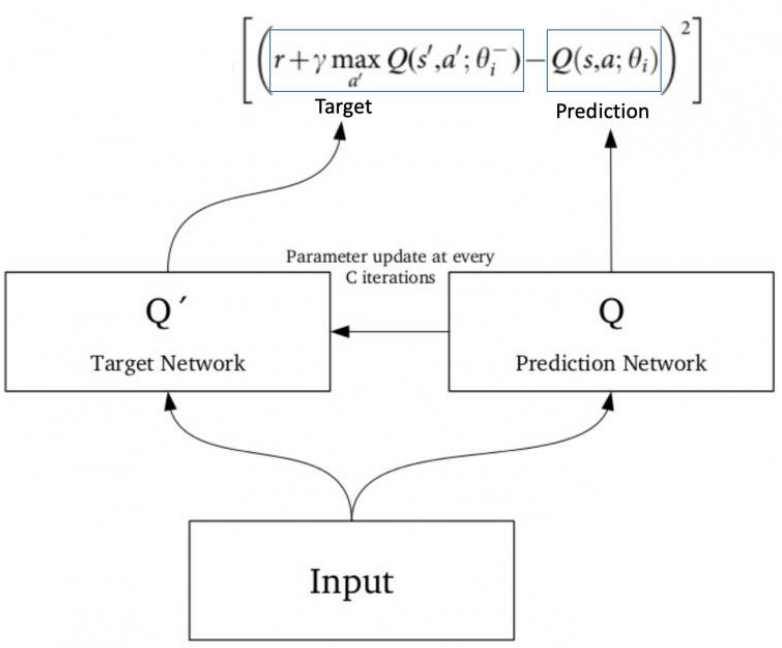
obtenu de la part de **l'équation de Bellman**, et ensuite calculer la différence entre le « target » et la valeur « y » pour obtenir l'erreur.

Important :

Un problème qui se pose dans le « Deep Q-Learning » est que l'équation de Bellman nous donne la valeur de Q (*s*, *a*) via Q (*s′*, *a′*). Cependant, il n'y a qu'un pas entre les deux états *s* et *s′*. Cela les rend si similaires qu'il est difficile pour un réseau de neurones de les distinguer. Lorsque nous effectuons des mises à jour de paramètres sur le réseau, afin de rapprocher Q (*s*, *a*) du résultat attendu.

Il existe une astuce, appelée **« target network »**, le principe est de garder une copie de notre réseau et qu'on l'utilise pour la valeur Q (*s′*, *a′*) dans l'équation de Bellman.

La figure ci-dessous nous montre le « target » et la valeur de prédiction « y » dans l'équation de Bellman, ainsi que l'utilité du « target network ».



Le « Deep Q-learning » a tendance à surestimer les valeurs de Q ce qui peut nuire aux performances d'entraînement. La cause de cet inconvénient est l'opération « max » de l'équation de Bellman. Donc les chercheurs on proposer un petit changement dans l'équation qui sera comme suite :

$$Q(s_t, a_t) = r_t + \gamma \max_a Q'(s_{t+1}, \arg\max_a Q(s_{t+1}, a))$$

Et cette nouvelle approche est appelée **« Double Deep Q-Learning »**.

Il existe d'autres inconvénients dans le « Deep Q-Learning » comme la grande taille de l'espace d'action, ou les actions continuent, qui peuvent ralentir la convergence du modèle. Cet inconvient peut être résolu par l'algorithme de **« Policy Gradient »** et **« Actor-Critic (A2C) »**

Ressources supplémentaires :

https://towardsdatascience.com/the-complete-reinforcement-learning-dictionary-e16230b7d24e

https://pylessons.com/CartPole-reinforcement-learning

https://developers.google.com/machine-learning/glossary/rl

https://cdn.hackernoon.com/hn-images/1*JJ3Dx4O3blc_haCUjv5Y5A.jpeg