

MODULE : IAD & AI

L'Intelligence Artificielle Distribuée & Agent Intelligent

Master 1 : Science de Données et Intelligence Artificielle

2023 - 2024

Plan de Présentation

- Apprentissage par Renforcement :

Définitions , Types d'Apprentissage, Comparaison, RL pour Agent ..

- Exploration & Exploitation.

- La Stratégie : ϵ -Greedy.

- Valeur d'Etat : $V(s)$.

- Valeur d'Action : $Q(s,a)$.

- L'Algorithme : Q-Learning.

- L'Algorithme : Deep Q-Learning.

- L'Algorithme : Policy Gradient.

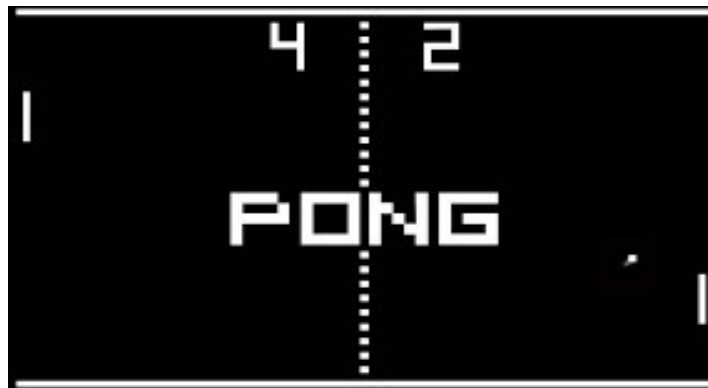
Deep Q-Learning (DQL)

- ▶ Le Deep Q-Learning (DQL) est une extension du Q-Learning traditionnel, qui utilise des réseaux de neurones profonds pour estimer les valeurs Q au lieu d'une table Q explicite.
- ▶ Cette approche permet de gérer des espaces d'états et d'actions continus ou de grande taille, ce qui rend le DQL plus adapté à des problèmes complexes tels que les jeux vidéo ou la robotique.



Exemple : Deep Q-Learning

- ▶ La cas d'un agent qui doit apprendre à jouer à un jeu vidéo simple, comme Pong.
- ▶ L'agent contrôle une raquette et doit renvoyer la balle tout en empêchant l'adversaire de marquer.
- ▶ L'agent peut déplacer la raquette vers le haut ou vers le bas à chaque étape.




Exemple : Deep Q-Learning

1. **Initialisation du réseau neuronal** : Initialiser un réseau de neurones profond qui prend l'état du jeu en entrée et produit une estimation des valeurs Q pour chaque action en sortie.
2. **Exploration vs Exploitation** : Utiliser une stratégie epsilon-greedy pour sélectionner les actions.
 - ▶ Au début, l'agent explorera davantage pour découvrir différentes stratégies, puis exploitera progressivement ses connaissances.
3. **Prétraitement des données** : Prétraiter les images du jeu pour les adapter à l'entrée du réseau de neurones (par exemple, redimensionnement, normalisation).



Exemple : Deep Q-Learning

4. **Interaction avec l'environnement** : À chaque étape, l'agent observe l'état du jeu, sélectionne une action en fonction de sa politique actuelle (**réception d'une récompense**).
 5. **Mise à jour du réseau neuronal** : Utiliser l'algorithme de **descente de gradient** pour mettre à jour les poids du réseau afin de **minimiser la différence entre les valeurs Q prédites et les valeurs Q cibles** (**avoir des target de l'état**).
 - ▶ Les valeurs Q cibles sont calculées en utilisant la formule de mise à jour Q-Learning, mais avec une petite modification pour stabiliser l'apprentissage, souvent appelée "**target network**".
 4. **Répéter** : Répéter les **étapes 3 à 5** pour un certain nombre d'itérations d'apprentissage.
- 

Architecture : Deep Q-Learning

- **Entrée** : Une image du jeu (exemple : 84x84 pixels en niveaux de gris).
- **Couche convolutive** : Appliquer plusieurs filtres convolutifs pour extraire les caractéristiques de l'image.
- **Couche entièrement connectée** : Prendre les caractéristiques extraites et estimer les valeurs Q pour chaque action possible (exemple : haut, bas).
- ✓ En continuant à entraîner ce réseau neuronal sur de nombreuses itérations, l'agent apprendra à jouer efficacement, en maximisant les récompenses reçues.

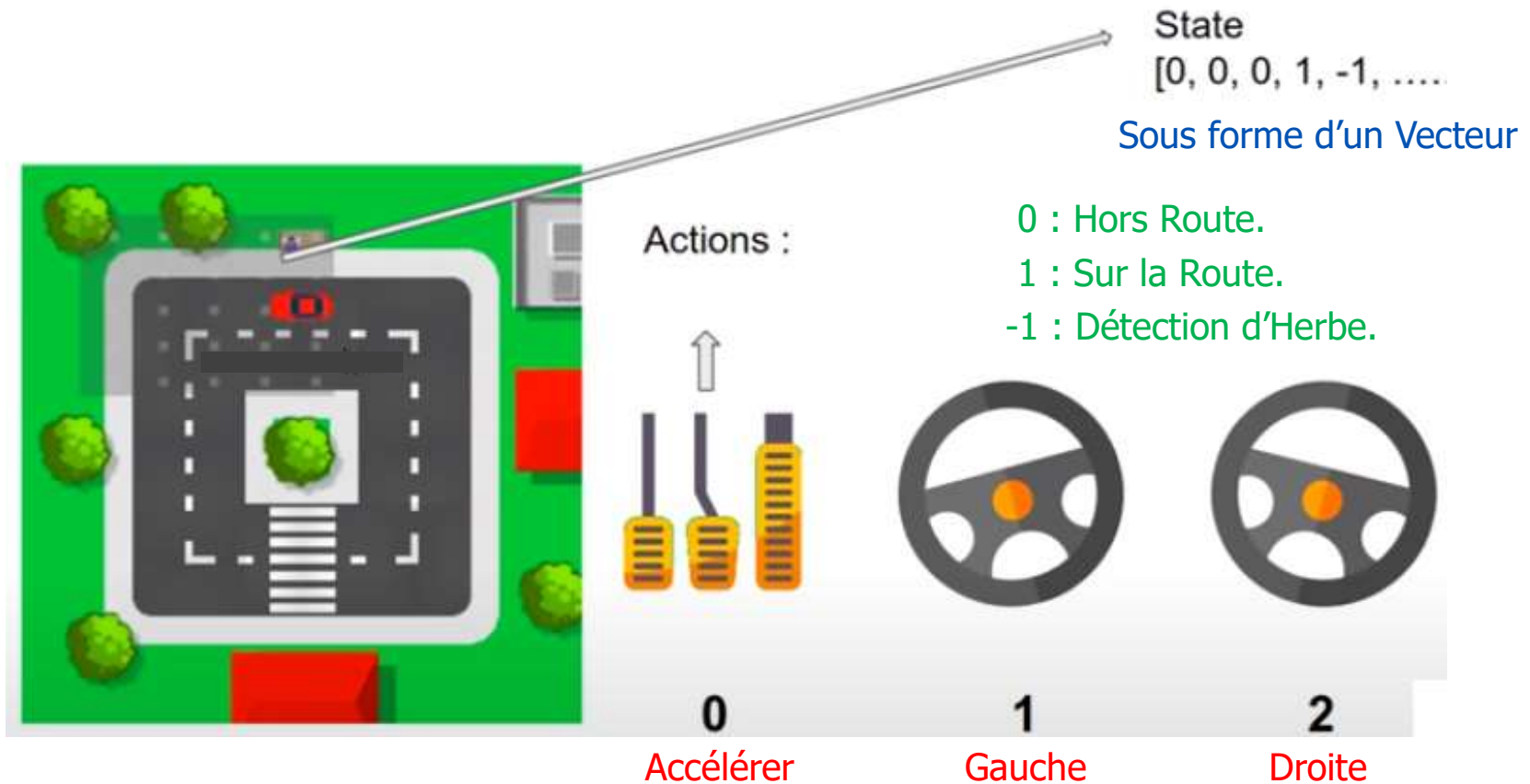


Exemple Illustratif

La Voiture Autonome



Exemple Illustratif



Problématique : Selon cet état, la Q-Table va s'agrandir d'une manière exponentielle.

- ✓ **L'idée :** Faire abstraction à travers des Réseaux de Neurones (RN), combinés à l'apprentissage par renforcement.

Trois Utilisations du même RN

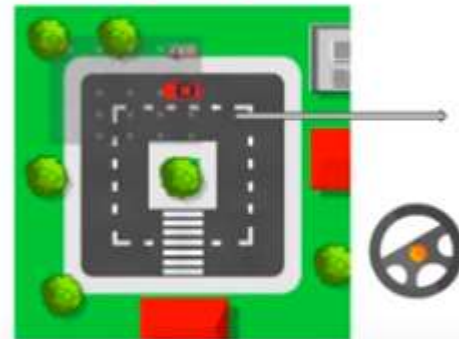
Même Etat :
Action 1



$$Q_{\theta}(s, a)$$

Espérance 1
RN 1

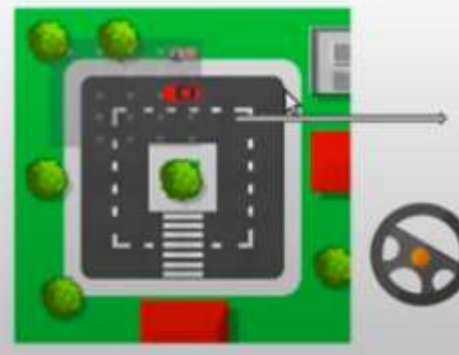
Même Etat :
Action 2



$$Q_{\theta}(s, a)$$

Espérance 2
RN 2

Même Etat :
Action 3

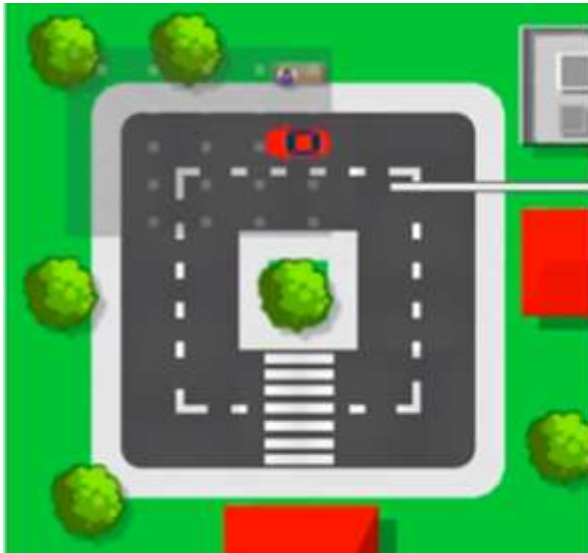


$$Q_{\theta}(s, a)$$

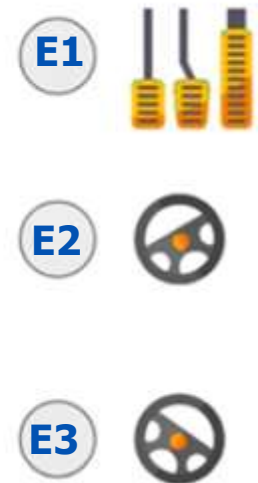
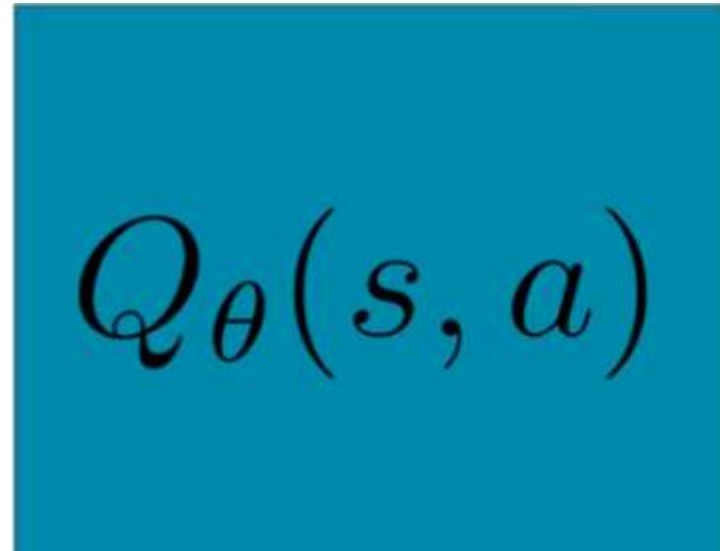
Espérance 3
RN 3

Optimisation en un seul RN

Une Seule
Entrée



Avec un Seul
Réseau de Neurone

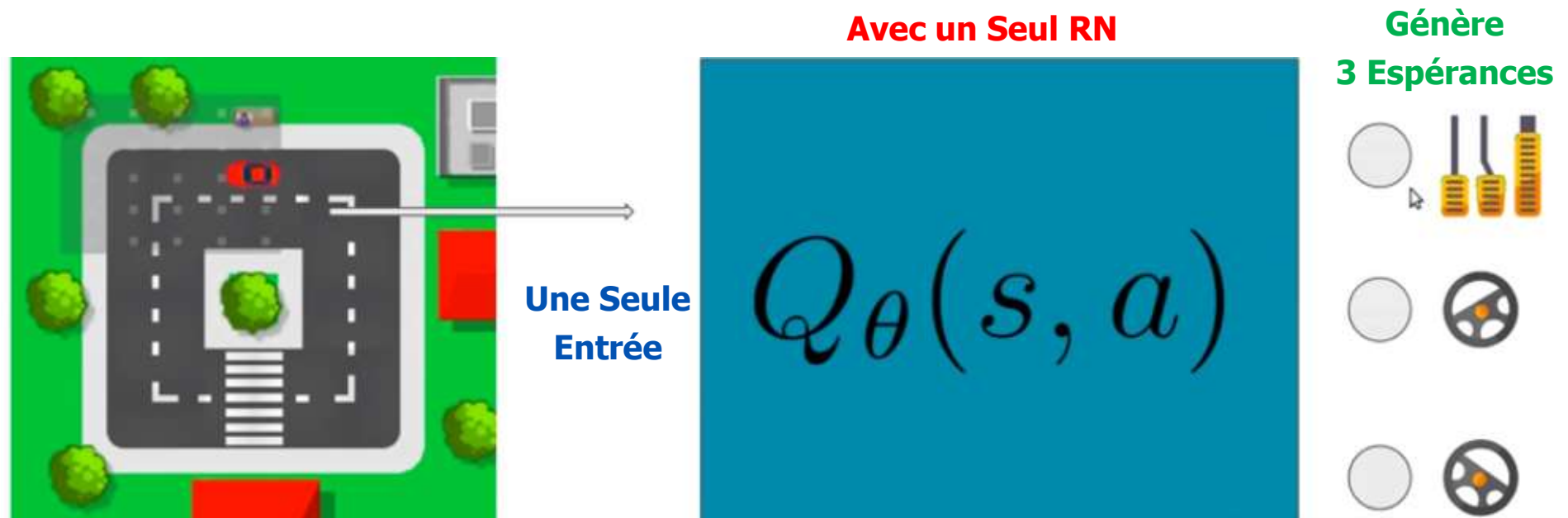


Génère
3 Espérances

Adopter une simple architecture du RN :
Une seule Couche Cachée (Représentée par la Q-Fonction).

Trouver l'Action Optimale

$$Q^*(s, a)^\pi = \max_\pi \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots | s_t = s, a_t = a, \pi]$$

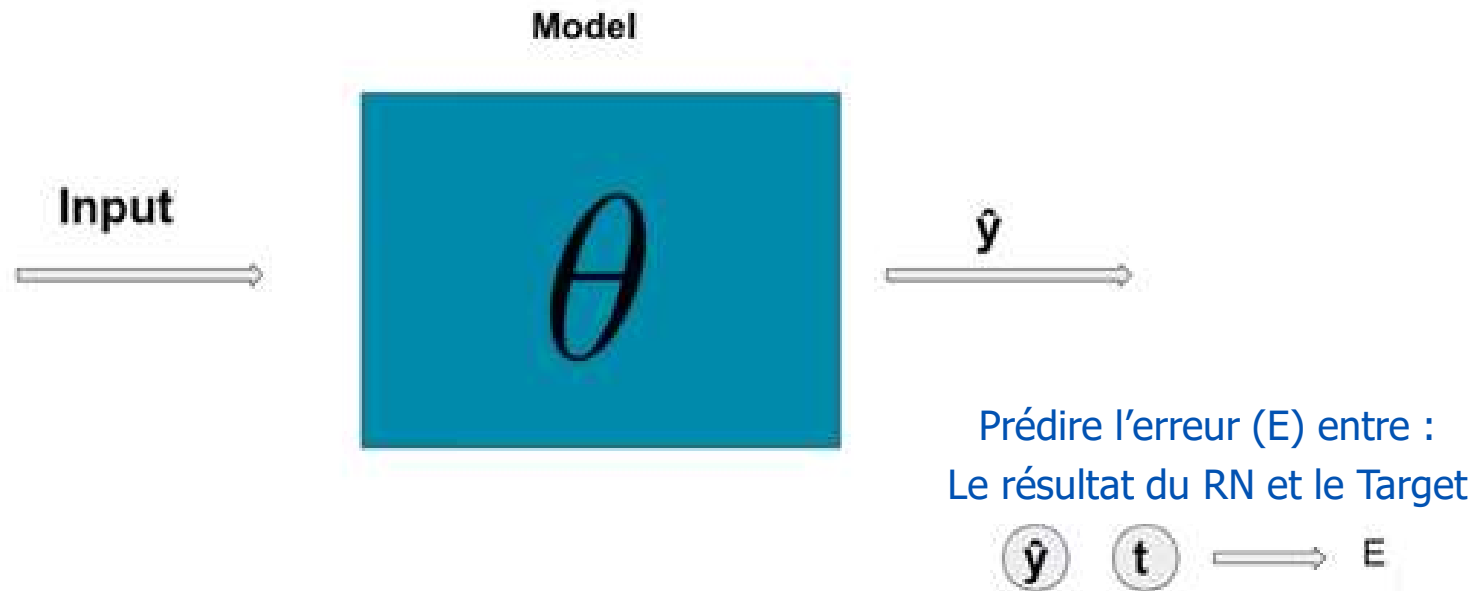


- ✓ **L'idée :** On cherche une moyenne d'espérance sur tous les épisodes.
Selon une politique π , pour trouver le comportement optimal.

Réduire l'erreur de prédiction

$$Q^*(s, a)^\pi = \max_\pi \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} + \dots | s_t = s, a_t = a, \pi]$$

- ✓ Tout en cherchant à réduire la valeur d'erreur **E**, entre les sorties prédites et les sorties réelles (*selon notre Dataset de l'environnement complexe*).



- ✓ *L'idée :*

Changer les poids des Réseaux de Neurones (RN).

Q-Learning vs Deep Q-Learning

❖ Q-Learning :

- 1. Méthode basée sur la table Q** : Le Q-Learning est une méthode d'apprentissage par renforcement basée sur une table appelée (Table Q). Cette table stocke les valeurs Q, qui représentent la qualité des actions dans un état donné.
- 2. Exploration-Exploitation** : Le Q-Learning utilise une stratégie d'exploration-exploitation pour découvrir et exploiter les meilleures actions. Il peut utiliser des méthodes telles que l'epsilon-greedy pour équilibrer l'exploration et l'exploitation.
- 3. Convient aux environnements discrets** : Il fonctionne bien dans les environnements où l'espace des états et des actions est discret et de petite taille.



Q-Learning vs Deep Q-Learning

❖ Deep Q-Learning (DQL) :

1. **Utilisation des réseaux de neurones profonds** : Contrairement au Q-Learning classique, le Deep Q-Learning utilise des réseaux de neurones profonds pour estimer les valeurs Q. Cela permet de gérer des espaces d'états et d'actions plus vastes et continus.
2. **Apprentissage de fonctions d'actions** : Au lieu de stocker explicitement les valeurs Q pour chaque paire état-action dans une table, le DQL apprend une fonction d'action $Q(s,a)$ approximative à l'aide d'un réseau de neurones.
3. **Gestion des problèmes continus** : Le Deep Q-Learning est plus adapté aux environnements où l'espace des états et des actions est continu ou de grande dimension.





***Merci pour votre
attention ..***