SITHIDEJ Clara
TOURAINE Florian

Mini-Projet : Apprentissage par renforcement

8INF867 : Fondamentaux de l'Apprentissage Automatique

UQAC

Automne 2025

2.1 Synthèse de notre compréhension de l'algorithme DQN

L'apprentissage par renforcement est composé d'un agent qui apprend à interagir avec un environnement en prenant des actions, en observant les conséquences de ces actions et en recevant des récompenses. Contrairement à l'apprentissage supervisé que nous avons vu dans les devoirs précédents, il n'existe pas de données étiquetées au départ. L'agent apprend progressivement par des cycles essais-erreurs dans le but de maximiser une récompense cumulée sur le long terme.

Parmi les algorithmes de l'apprentissage par renforcement, on retrouve le Q-learning, qui consiste à apprendre une fonction de valeur d'action, notée *Q(s,a)*. Cette fonction représente la qualité d'une action *a* prise dans un état *s*, autrement dit, l'espérance de la récompense cumulée que l'agent peut obtenir en suivant une politique donnée à partir de cet état. Le Q-learning classique met à jour cette fonction en se reposant généralement sur une table *Q*. Cette approche devient rapidement impraticable lorsque l'espace des états est grand ou continu. Afin de pallier cette limitation, l'algorithme Deep Q-Network (DQN) a été introduit par Mnih et al. (2015). Le principe du DQN est d'utiliser un réseau de neurones profond pour approximer la fonction *Q(s,a)* au lieu d'une table qui en donnerait directement toutes les valeurs. Le réseau prend en entrée une représentation de l'état et produit en sortie une valeur Q pour chaque action possible. L'agent choisit alors l'action associée à la plus grande valeur Q.

Cependant, l'utilisation directe d'un réseau de neurones dans le Q-learning pose des problèmes de stabilité et de divergence lors de l'entraînement. Pour résoudre ces problèmes, le DQN introduit deux mécanismes essentiels. Le premier est l'experience replay. Les interactions de l'agent avec l'environnement sont stockées sous forme de transitions (s, a, r, s') dans une mémoire. Lors de l'apprentissage, des mini-lots de transitions sont tirés aléatoirement depuis cette mémoire, ce qui permet de casser les corrélations temporelles entre les données et d'améliorer la stabilité de l'apprentissage. Le second mécanisme est l'utilisation d'un réseau cible (target network). En plus du réseau principal, un second réseau est utilisé pour calculer les valeurs cibles lors de la mise à jour des poids. Ce réseau cible est mis à jour moins fréquemment que le réseau principal, ce qui permet de réduire les oscillations et d'assurer une convergence plus stable.

Enfin, l'exploration de l'environnement est généralement assurée par une politique ε-greedy. Cette stratégie consiste à choisir une action aléatoire avec une probabilité ε afin d'explorer l'environnement, et à choisir l'action optimale selon le réseau avec une probabilité 1 - ε. La valeur de ε décroît progressivement au cours de l'apprentissage, permettant à l'agent de passer d'une phase d'exploration à une phase d'exploitation de la politique apprise.

Ainsi, le Deep Q-Network combine les principes du Q-learning avec la puissance des réseaux de neurones profonds afin de permettre l'apprentissage dans des environnements complexes et de grande dimension. Grâce à l'utilisation de l'expérience replay et du réseau cible, le DQN constitue une approche efficace et stable pour résoudre des problèmes d'apprentissage par renforcement.

Voici les principales références que nous avons utilisé :

- Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529–533.
- Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd ed.). MIT Press.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

2.2 Contexte d'application du DQN et justification du choix

Dans le cadre de ce projet, nous avons appliqué l'algorithme Deep Q-Network (DQN) au problème du GridWorld. Il s'agit d'un environnement matriciel où un agent doit naviguer d'une position de départ vers un objectif précis tout en contournant des obstacles fixes. Ce type de simulation constitue un environnement idéal pour l'apprentissage par renforcement, car il repose sur une interaction directe et continue entre un agent situé et son environnement.

Dans le GridWorld, l'agent doit prendre des décisions séquentielles à chaque étape, comme se déplacer vers le Haut, le Bas, la Gauche ou la Droite. L'apprentissage par renforcement est ici fondamental car l'agent ne possède pas d'historique de données préalable pour guider ses choix, contrairement aux méthodes supervisées classiques. Il doit explorer l'espace pour maximiser une récompense cumulée, souvent représentée par l'atteinte rapide de la cible et l'évitement des pénalités liées aux collisions avec les obstacles.

Le choix du DQN est particulièrement justifié car il permet à l'agent de découvrir progressivement, par essais et erreurs, la politique optimale sans stratégie connue à l'avance. Bien que le GridWorld puisse paraître simple, la complexité augmente avec la taille de la grille et la disposition des obstacles. L'utilisation d'un réseau de neurones profond permet d'approximer efficacement la fonction de valeur d'action (Q) et de généraliser l'apprentissage à des positions ou configurations de grille que l'agent n'a pas nécessairement rencontrées de manière exhaustive durant son entraînement.

2.3 Architecture de l'agent apprenant, états, actions et mécanisme de récompense

L'objectif de notre agent-apprenant est d'apprendre une politique optimale permettant de naviguer dans la grille afin d'atteindre la cible tout en maximisant la récompense cumulée. Notre approche repose sur l'algorithme Deep Q-Network (DQN) classique, combinant les éléments fondamentaux suivants:

- un réseau de neurones profond pour approximer la fonction de valeur d'action *Q(s, a)*
- une stratégie d'exploration de type ε-greedy
- un replay buffer pour casser la corrélation temporelle entre les transitions
- un réseau cible (target network) pour stabiliser l'apprentissage

Notre agent interagit avec un environnement personnalisé, qui encapsule la logique de la grille (murs, cases vides, but) et fournit une interface standardisée pour l'apprentissage par renforcement via les fonctions (reset, step).

Nous avons choisi une représentation de l'état du DQN comme un vecteur de dimension 25, défini comme une représentation flatten de la grille

Cette représentation permet à l'agent de disposer d'informations essentielles sur sa position, les menaces immédiates et les opportunités de tir, tout en conservant un espace d'états de taille réduite. Quant à elles, les actions sont au nombre de quatre :

- 0 : se déplacer vers le haut
- 1 : se déplacer vers le bas
- 2 : se déplacer vers la gauche
- 3 : se déplacer vers la droite

Ces actions sont directement appliquées au jeu via une fonction dédiée, sans passer par le système de gestion du clavier de Pygame. La fonction de récompense joue un rôle central dans l'apprentissage de l'agent. Elle a été conçue afin d'encourager les comportements souhaités tout en pénalisant les actions inefficaces ou dangereuses. Les principaux termes de la récompense sont :

- une récompense positive importante lorsque l'agent arrive au but
- une pénalité forte si il rencontre un obstacle et tombe dedans

La fonction *Q* est approximée par un réseau de neurones entièrement connecté implémenté avec PyTorch. Le réseau prend en entrée l'état courant et produit une valeur Q pour chacune des actions possibles. L'apprentissage repose sur la minimisation de l'erreur quadratique moyenne entre la valeur *Q* prédite pour l'action choisie et la cible TD calculée à partir de la récompense observée et de la valeur maximale estimée par le réseau cible pour l'état suivant. Les transitions sont stockées dans un replay buffer de taille finie, et l'entraînement est effectué par mini-lots afin d'améliorer la stabilité et l'efficacité de l'apprentissage. Nous utilisons une stratégie ε-greedy avec un taux d'exploration initial élevé (ε = 1), décroissant progressivement jusqu'à une valeur minimale. Cette approche permet à l'agent d'explorer largement l'espace d'actions au début de l'entraînement, puis de privilégier progressivement l'exploitation de la politique apprise. L'entraînement est réalisé sur plusieurs épisodes indépendants. À la fin de chaque épisode, le réseau cible est mis à jour et la récompense cumulée est enregistrée afin d'analyser l'évolution des performances de l'agent. Cette première implémentation du DQN constitue une base fonctionnelle, permettant déjà à l'agent de se déplacer et d'arriver à de bons résultats.

2.5 Mesures des performances du DQN

L'évaluation de notre agent-DQN repose sur deux indicateurs fondamentaux : le taux de succès et le temps d'entraînement. Dans le cadre de notre expérimentation sur un GridWorld de 5x5, les résultats obtenus sont particulièrement élevés.

2.5.1 Analyse du Taux de Succès (Accuracy)

Le taux de succès, calculé comme le ratio entre le nombre d'épisodes réussis (atteinte de l'objectif) et le nombre total d'épisodes, atteint une valeur proche de 100%.

Influence de la taille de la grille :
Avec un espace d'états restreint de seulement 25 cases (5×5), l'agent parvient rapidement à cartographier l'intégralité de l'environnement.

Volume d'entraînement :
L'utilisation de 1000 épisodes (epochs) constitue un sur-apprentissage volontaire pour ce type de grille. Cela permet de garantir que l'exploration via la stratégie ϵ-greedy a été exhaustive et que la fonction de valeur Q a parfaitement convergé vers la solution optimale

2.5.2 Analyse du Temps d'Entraînement

Le temps d'entraînement global a été mesuré afin d'évaluer l'efficacité de l'algorithme.

Rapidité de convergence :
Bien que l'entraînement soit plus long que celui d'un modèle supervisé, la simplicité de la grille permet à l'agent d'apprendre une politique stable en un temps très court.

2.7 Mesures des performances du MLP

Le MLP supervisé s'est entraîné sur des données synthétiques pour faire de la classification. Il apprend en fonction de la position de l'agent à prendre la bonne décision. Les données ont été créés à l'aide de la position aléatoire de l'agent ainsi que le meilleur chemin calculé avec un A* dont on prend la première décision.

2.7.1 Analyse du Taux de Succès (Accuracy)

Le MLP est particulièrement optimisé pour ce problème ce qui donne une accuracy proche de 100% pour la classification.

2.7.2 Analyse du Temps d'Entraînement

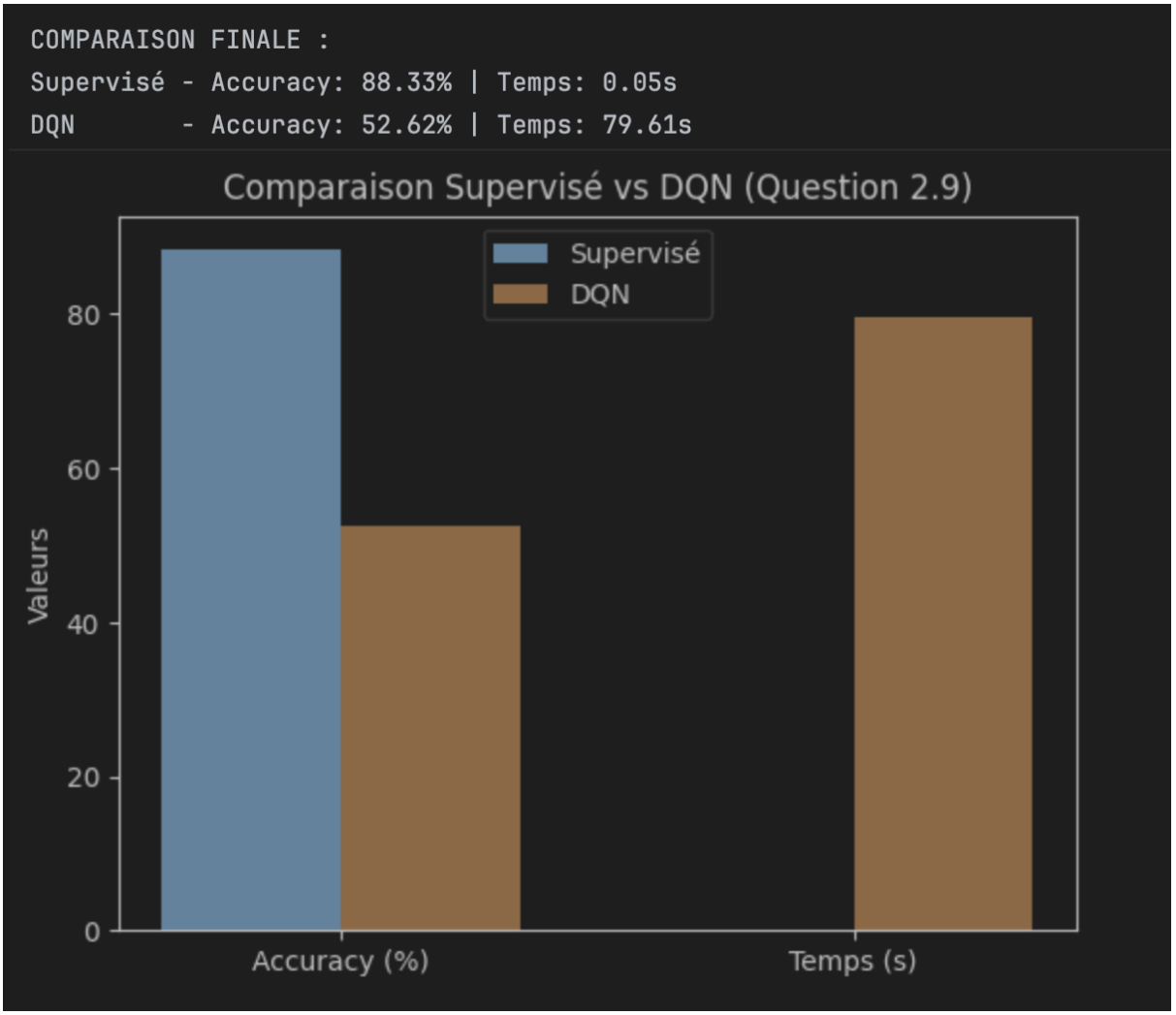
Le temps d'entraînement est très court pour 100 époques.

2.9 Comparer les deux modèles d'agent

Dans cette section, nous comparons l'agent-supervisé (MLP) entraîné sur un dataset d'expert à l'agent-DQN adapté pour la tâche de classification. Les deux modèles ont été testés sur une grille identique de 5×5 avec les mêmes observations d'entrée (coordonnées [x,y]).

L'apprentissage supervisé s'est révélé être l'approche la plus efficace pour ce problème spécifique de classification. Sa supériorité s'explique par le fait qu'il reçoit directement le label optimal pour chaque état. À l'inverse, l'agent-DQN doit passer par une phase d'exploration (ϵ-greedy) où il teste des actions sous-optimales pour découvrir leur valeur de récompense, ce qui ralentit sa convergence. De plus, l'apprentissage supervisé s'est montré très rapide par rapport au DQN.

```
COMPARAISON FINALE :
Supervisé - Accuracy: 88.33% | Temps: 0.05s
DQN       - Accuracy: 52.62% | Temps: 79.61s
```



Conclusion

Ce projet nous a permis d'explorer deux paradigmes fondamentaux de l'intelligence artificielle appliqués à la navigation dans un GridWorld : l'apprentissage par renforcement profond (DQN) et l'apprentissage supervisé.

L'implémentation de l'agent-DQN a démontré la puissance des réseaux de neurones pour approximer des fonctions de valeur dans un espace d'états. Nous avons constaté que la stabilité de cet apprentissage repose sur des mécanismes critiques tels que le *Replay Buffer*.

L'étude comparative a mis en évidence que pour un environnement fixe et documenté, l'approche supervisée offre une précision et une rapidité d'exécution supérieures. Toutefois, l'apprentissage par renforcement reste la solution de choix pour les problèmes complexes où aucune solution experte n'est disponible.

En conclusion, ce travail souligne l'importance du choix de l'architecture et de la méthode d'apprentissage en fonction des données disponibles et de la nature de l'environnement. Les excellents résultats obtenus sur notre grille 5×5 valident notre approche, tout en ouvrant la voie à des extensions vers des environnements plus vastes et dynamiques.

Quelques améliorations peuvent être apportées, comme l'entraînement du DQN sur une grille plus grande ou bien sur une grille qui évolue au cours des itérations.y