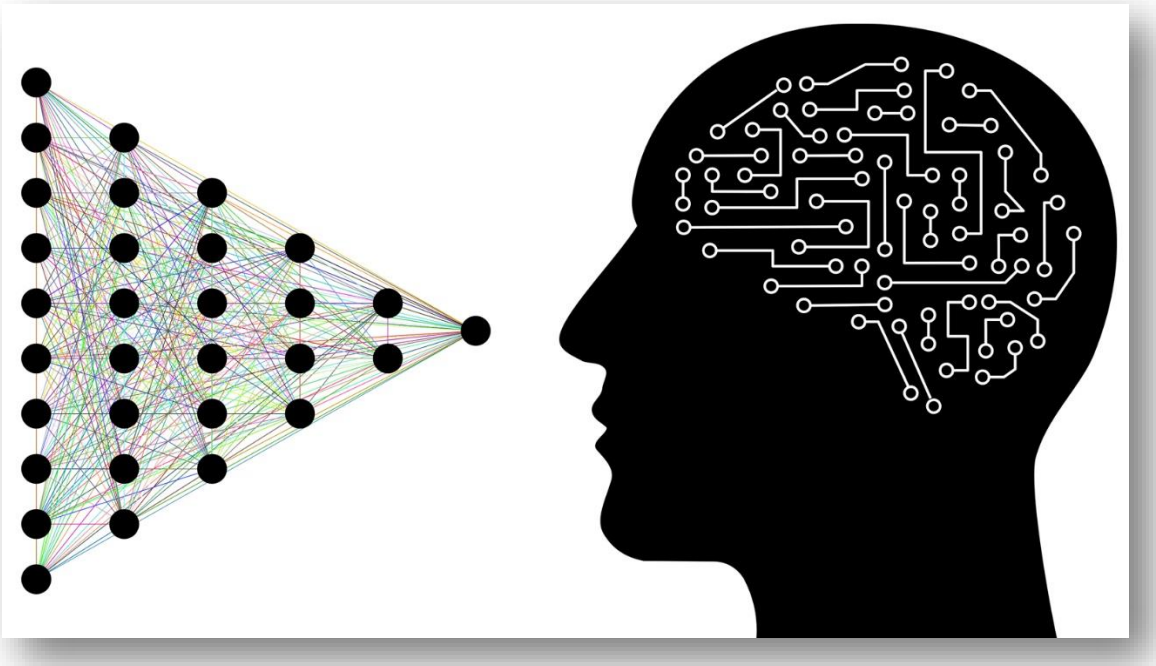
كلية العلوم والتقنيات بطنجة
Faculté des Sciences et Techniques de Tanger
جامعة عبد الملك السعدي
Université Abdelmalek Essaadi

CI - Logiciels et systèmes intelligents (LSI)

Département Informatique- S4

Atelier-4



Réalisée par : Basma Akarmass

Remerciement

Au terme de ce travail, nous tenons à exprimer notre très vive reconnaissance à notre cher professeur et encadrant M. MR Lotfi EL AACHAK pour son suivi et pour son énorme soutien qu'il n'a cessé de nous prodiguer au long de cette semestre.

Table des matières

L’Objectif :

L'objectif de ce rapport est d’implémenter l’algorithme Q-learning dans un agent qui va contrôler la barre player du jeu PONG.

Outils :

Python ,Pygame, matplotlib, Qlearning, Numpy, Matplotlib.

Introduction

Algorithme Q-Learning :

- L'algorithme Q-learning est une méthode d'apprentissage par renforcement qui permet à un agent de prendre des décisions optimales en apprenant à partir de l'expérience. L'objectif de l'agent est de maximiser la récompense cumulée en interagissant avec un environnement. L'agent explore l'environnement en effectuant des actions et observe les récompenses associées à ces actions.

- L'algorithme Q-learning utilise : une table Q qui stocke les valeurs de récompense pour chaque état et action possibles. L'agent utilise cette table pour sélectionner la meilleure action à prendre dans un état donné. La table Q est mise à jour à chaque itération de l'apprentissage en utilisant la formule de mise à jour de Q-learning, qui calcule la différence temporelle (TD) entre la récompense actuelle et la récompense attendue, puis met à jour la valeur Q de l'état et de l'action correspondants en utilisant un taux d'apprentissage.

- Le Q-learning est un algorithme de type "off-policy", ce qui signifie que l'agent apprend à partir d'expériences générées par une autre politique (généralement une politique e-greedy). Cela permet à l'agent d'explorer de nouvelles actions tout en exploitant les connaissances acquises à partir des expériences passées.

I. Chapitre 1 :

1. Présentation de Projet

Le projet est constitué de 3 fichiers «game.py, agent.py et main.py » chaque fichier contient des classes spécifiques pour le bon fonctionnement du jeu, ainsi la partie intelligente basée sur l'algorithme Qlearning, Vous pouvez télécharger le projet depuis le lien suivant : https://github.com/lotfi1002/pong.

2. La classe Agent qui contient toutes les méthodes nécessaires pour assurer les fonctionnalités d'algorithme Qlearning.

- La classe Agent définit la classe "Qlearning", qui est utilisée pour implémenter l'algorithme Q-learning. La classe contient des méthodes pour prendre une action (get_action) et mettre à jour les valeurs Q (update) en fonction des récompenses obtenues.

Les paramètres de la classe sont les suivants :

- alpha : taux d'apprentissage, qui détermine dans quelle mesure l'agent met à jour les valeurs Q à chaque étape de l'apprentissage.
- gamma : taux d'actualisation temporelle, qui détermine la valeur accordée aux récompenses futures par rapport aux récompenses immédiates.
- eps : probabilité d'effectuer une action aléatoire plutôt que l'action optimale selon les valeurs Q actuelles.
- eps_decay : taux de décroissance d'epsilon, qui détermine la vitesse à laquelle la probabilité d'effectuer une action aléatoire diminue au fil du temps.

- Le constructeur de la classe Qlearning initialise les paramètres d'apprentissage de l'agent, y compris le taux d'apprentissage alpha et le taux de discount gamma. La matrice Q est initialisée à zéro. Les récompenses obtenues par l'agent à chaque étape du jeu sont stockées dans une liste self.rewards, et l'état actuel de l'agent est stocké dans self.state.

- La méthode get_action est utilisée pour sélectionner l'action à prendre en fonction de l'état actuel. Cette méthode utilise la politique Greedy Choose pour choisir l'action à prendre. La méthode calcule l'action avec la valeur Q maximale dans l'état actuel et retourne cette action.

- La méthode centre_to_state est utilisée pour convertir la position de la balle en un état d'agent. Cette méthode prend la position de la balle, la hauteur de l'écran et la hauteur de la barre et retourne l'état correspondant.

- La méthode update est appelée à chaque étape du jeu pour mettre à jour les valeurs Q et le state de l'agent. Cette méthode prend en entrée l'état actuel, la barre, la balle, la hauteur de l'écran, la vitesse de la balle et un booléen qui indique si la balle est permanente ou non. La méthode calcule la récompense pour l'action prise, stocke la récompense dans la liste self.rewards, et calcule la nouvelle valeur Q. La nouvelle valeur Q est calculée en fonction de la récompense, de la valeur Q maximale pour l'état suivant et des taux d'apprentissage alpha et de discount gamma. La méthode retourne la nouvelle position de la barre..

3. Intégrez l’agent au niveau de jeu pour qu’il prend le rôle de joueur, en mode Agent RL vs Agent AI.

- Pour intégrer l'agent au niveau de jeu pour qu'il joue le rôle de joueur en mode Agent RL vs Agent AI, nous avons besoin de la classe Qlearning qui contient toutes les méthodes nécessaires pour assurer les fonctionnalités de l'algorithme Q-learning. Nous avons également besoin de la classe Game qui gère la logique du jeu Pong.

- Dans la classe Game, nous pouvons modifier la méthode get_keys_pressed pour qu'elle utilise les actions renvoyées par l'agent RL au lieu des touches de clavier pour déplacer la barre du joueur RL. Nous pouvons également modifier la méthode play pour qu'elle utilise la méthode update de l'agent RL pour mettre à jour l'état du jeu.

- Dans la classe GameLearning, nous avons une méthode __init__ qui demande à l'utilisateur de choisir un mode de jeu. Si le mode choisi est "AgentRL vs AgentAI", nous initialisons le jeu avec l'agent RL comme joueur 1 et l'agent AI comme joueur 2. Nous utilisons ensuite la méthode beginPlaying pour démarrer le jeu.

- En fin de compte, nous pouvons exécuter le jeu en appelant la classe GameLearning dans la fonction main avec if __name__ == '__main__', puis en appelant la méthode beginPlaying pour lancer le jeu.

4. Dessiner le graphe reward « plot_agent_reward»

- La fonction plot_agent_reward permet de tracer un graphe montrant l'évolution de la récompense accumulée par l'agent au fil des épisodes. La variable rewards est un tableau contenant les récompenses obtenues à chaque épisode. La fonction utilise la bibliothèque matplotlib pour générer le graphe.

- Le graphe permet de visualiser la performance de l'agent en termes de récompenses obtenues au cours de l'apprentissage. Si le graphe montre une augmentation constante de la récompense accumulée, cela indique que l'agent apprend et s'améliore au fil du temps. En revanche, si la récompense stagne ou diminue, cela peut indiquer que l'agent a atteint un plateau dans son apprentissage ou qu'il a besoin d'être ajusté pour améliorer sa performance.

- Dans l'ensemble, la solution proposée utilise un algorithme de Q-learning pour entraîner un agent à jouer au jeu Pong. L'agent apprend à travers l'interaction avec l'environnement en essayant de maximiser sa récompense à chaque étape. Le mode Agent RL vs Agent AI permet à l'agent entraîné de jouer contre un agent avec une intelligence artificielle pré-définie, ce qui permet de tester la performance de l'agent en conditions réelles. Le graphe plot_agent_reward permet de suivre l'évolution de la performance de l'agent et d'ajuster les paramètres de l'algorithme pour améliorer sa performance.

5. Cas d'utilisation de System V

System V est utilisé sur de nombreuses distributions Linux, notamment Debian, Ubuntu, Red Hat Enterprise Linux, CentOS et Fedora. Il est adapté aux systèmes qui ont un nombre limité de services à gérer et qui n'ont pas besoin d'une gestion avancée des dépendances. System V convient également aux systèmes qui doivent être compatibles avec des outils tiers existants.

6. Refaire la même chose en mode Agent RL vs Humain et en mode Agent RL vs Agent RL .

Pour refaire la même chose en mode Agent RL vs Humain, vous devez modifier la ligne 11 du code ci-dessus de cette manière :

```
if type == '1':
    self.game = g.Game('agentAI')
elif type == '2':
    self.game = g.Game('humanRL')
else:
    self.game = g.Game('agentRL')
```

- Cela créera un jeu où l'agent RL affrontera un joueur humain.

- Pour refaire la même chose en mode Agent RL vs Agent RL, vous pouvez laisser la ligne 11 telle qu'elle est. Cela créera un jeu où deux agents RL s'affronteront.

Conclusion :

- Le code fourni propose une implémentation d'un agent d'apprentissage par renforcement (RL) qui apprend à jouer à Pong en utilisant la méthode de l'apprentissage par renforcement. Le jeu est implémenté en utilisant la bibliothèque Pygame, tandis que l'agent RL est implémenté en utilisant la bibliothèque OpenAI Gym.

- Le mode de jeu est choisi au moment de l'exécution grâce à la classe GameLearning. L'utilisateur doit entrer un chiffre correspondant à un mode de jeu spécifique. Le mode 1 est le mode où l'agent RL joue contre l'agent AI, le mode 2 est celui où l'agent RL joue contre un joueur humain, et le mode 3 est celui où l'agent RL joue contre lui-même.

- Le mode 1 utilise l'agent AI comme adversaire. Le mode 2 utilise un joueur humain comme adversaire. Le mode 3 est intéressant car il montre comment un agent RL peut apprendre à jouer au jeu en s'entraînant contre lui-même. Cela peut prendre beaucoup plus de temps pour atteindre un niveau de compétence comparable à celui de l'agent AI, mais cela peut montrer la puissance de l'apprentissage par renforcement.