Université De Lille
Faculté des Sciences et Technologies
Département de Mathématiques

Projet d'apprentissage autonome

# Une IA pour le jeu de Puissance 4

***Réalisé par :***

- Abdelmonssif *Oufaska*
- Kamal *Samnouni*

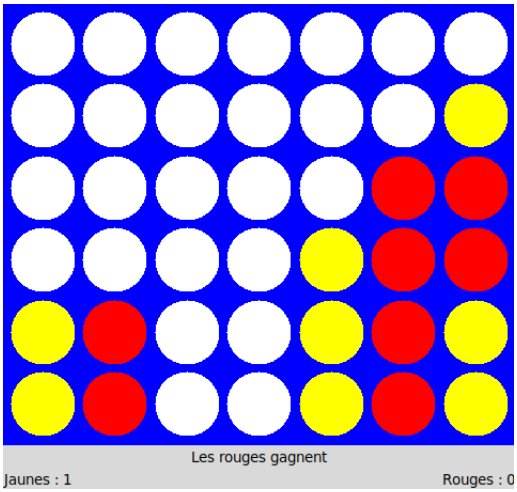
***Enseignant :***

- *Prof.* Nicolas *Wicker*

**Année Universitaire :**
2020/2021
20 Janvier 2021

# *Jeu de Puissance 4*

Le jeu de Puissance 4 est un jeu de stratégie à deux joueurs dans lequel le plateau est composé de sept colonnes (verticales), chacune disposant de six emplacements. Chaque joueur dispose de jetons d'une couleur donnée. Lors d'une partie, les joueurs placent successivement un pion de leur couleur dans l'une des colonnes. La partie s'arrête dès que l'un des joueurs a réussi à aligner (horizontalement, verticalement ou en diagonale) quatre pions de sa couleur, ce joueur est alors le gagnant. La grille de jeu est la suivante :




L'objectif de ce projet est d'apprendre à une IA (Intelligence Artificielle ) à jouer en utilisant l'apprentissage par renforcement. La méthode utilisée ici est le Q-learning avec typiquement 42 neurones en entrée et 7 neurones en sortie et 30 neurones sur la couche cachée. Au début, l'ordinateur apprendra en jouant un nombre de parties suffisamment grand contre lui-même. Ainsi, il y aura deux modes de fonctionnement du programme, un mode où il apprend et un autre mode où il est possible de l'affronter.

# Q-Learning

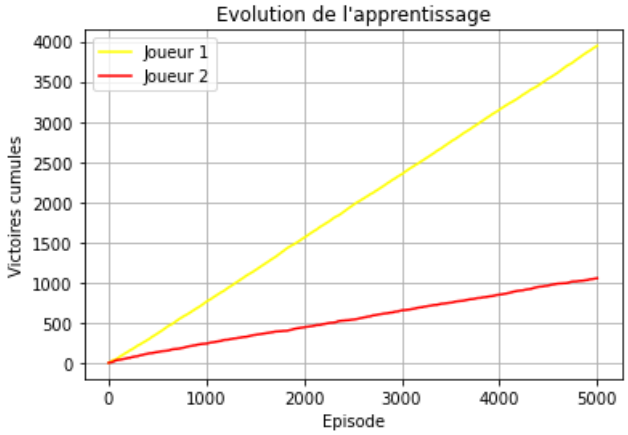
Le principe de cette méthode repose sur le calcul d'une table Q(state, action) qui pour un état « state » de la grille, donne le gain que le joueur peut espérer avoir lorsqu'il choisit l'action « action » . Ainsi lorsque c'est le tour de l'IA et que celle-ci se trouve dans l'état « state », elle se réfère à la table pour trouver l'action « action » qui maximise son espérance de gain Q(state, action).

Pour créer l'environnement du jeu pour chaque instant t du jeu, on a utilisé deux fonctions suivantes du type booléen :
– colonne_pleine : qui vérifiée si on a la possibilité de jouer une colonne donnée.
– alignements_troues : qui vérifiée s'il existe un alignement ( horizontalement, verticalement ou en diagonale) de quatre pions de même couleur

Initialement, notre IA (Agent 1) ne connaît aucune stratégie, il est donc trivial de gagner contre elle. Pour jouer de manière optimale, l'IA doit toujours sélectionner l'action « action » ( parmi les numéros 1,..,7 qui représentent les numéros des colonnes) qui maximise Q(state, action) lorsqu'elle se trouve dans l'état « state ». Mais comme initialement la table est très creuse, il est nécessaire de laisser l'IA explorer un peu librement son environnement. On choisit 5000 parties pour la partie apprentissage. Pour les récompenses on fixe le poids de chaque case comme suit :

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 4 | 5 | 7 | 5 | 4 | 3 |
| 4 | 6 | 8 | 10 | 8 | 6 | 4 |
| 5 | 8 | 11 | 13 | 11 | 8 | 5 |
| 5 | 8 | 11 | 13 | 11 | 8 | 5 |
| 4 | 6 | 8 | 10 | 8 | 6 | 4 |
| 3 | 4 | 5 | 7 | 5 | 4 | 3 |

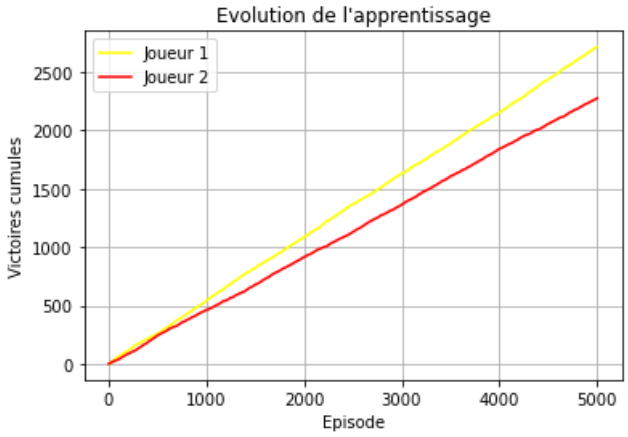
On a trouvé ces poids des cases du jeu de la puissance 4 sur Internet, en utilisant la théorie des poids. (Pour voir la source Ici)
On va entraîner notre IA contre plusieurs adversaires :
• Un adversaire aléatoire qui joue de manière aléatoire.
Le graphique ci-dessous représente le nombre cumulé de fois où notre IA gagne contre l'adversaire aléatoire en jaune, et en rouge le nombre de fois où l'adversaire aléatoire gagne contre l'IA :




On remarque que notre IA gagne souvent contre l'adversaire aléatoire le taux du victoire est de 80% sur 5000 parties. Donc elle semble que notre IA a appris des stratégies en chaque épisode.

• Elle-même : l'IA affronte un adversaire qui joue selon sa propre stratégie.

Le graphique ci-dessous représente le nombre cumulé de fois où notre IA gagne contre elle-même en jaune :




On remarque que notre IA gagne contre elle-même un peu de partie sur 5000 partie. Par ce qu'il connaît déjà la stratégie du jeu de son adversaire. On peut expliquer la perte contre elle-même par le choix d'une action au hasard avec une probabilité uniforme$[0, 1]$ inférieur à $\epsilon = 0.1$. • Un adversaire réel : un humain expérimenté.

Quand on affronte notre IA contre un joueur réel, elle semble qu'elle a pas appris assez de stratégies pour gagner contre un adversaire expérimenté.

Vous pouvez changer dans le fichier "MOTEUR" le paramètre "ALEATOIRE" qui prend "True" pour que l'IA joue contre l'adversaire aléatoire et "False" pour que l'IA joue contre elle-même.