Franck CARON
M1GIL

Mini-Projet
Apprentissage Automatique

Réponses aux questions

Les bases et le mode simple

Q] L'ordinateur peut faire des erreurs évidentes lors du dernier tour. A l'aide du code, expliquez pourquoi.

Dans le code : En mode « easy », un tour du jeu par l'ordinateur (CPU) est réalisé par la méthode playEasy de la classe CPUPlayer. Le corps de cette méthode montre qu'elle ne fait que réaliser un appel à la méthode playRandom, donnant un nombre quelconque de coups à jouer.
La probabilité de choix étant uniforme (1/nb de possibilités)

Explication :

Défini en mode de jeu "easy", a décision de jeu du CPU est alors prise de façon purement aléatoire, et ne suit donc aucune logique, notamment celle qui permettrait de se placer dans une position gagnante.

De ce fait, il se peut que lors de tours où le CPU peut se placer en position de victoire, en ne laissant qu'un bâton lors de l'avant-dernier tour, ce dernier peut en laisser un nombre différent, laissant ainsi la victoire au joueur adverse.

Mode intermédiaire

Q] Est-ce de l'apprentissage ? Expliquez.

Dans le code : On fixe un traitement particulier de l'IA intermédiaire via la méthode playMedium de la classe CPUPlayer. Dans le cas, où le nombre de bâtons restants est entre 2 et 4. On renvoie alors l'entier (nb de bâtons restant) – 1, laissant ainsi un bâton au joueur adverse.

Explication :

Par rapport à son implémentation, le comportement de l'IA intermédiaire se rapproche d'un apprentissage par explication. L'algorithme utilisé pour l'explication serait alors :

```
JEU_INTERMEDIAIRE(n : entier) =
        SI 2 ≤  n ≤  4 ALORS
                RENVOYER n – 1
        SINON
                RENVOYER JEU_ALEATOIRE(n)
        FSI
FIN
```

Cet algorithme est simpliste, et ne donne une réponse déterministe qu'à 3 cas sur les 15 disponibles, mais définit en soi un premier algorithme pour donner une bonne stratégie au CPU.

Apprentissage

Q] Nous allons maintenant laisser l'ordinateur jouer contre lui-même en mode "hard". Comment s'appelle cette méthode et pourquoi l'utilise-t-on ?

Explication :

La méthode utilisée est l'apprentissage par renforcement, elle est employée afin de renforcer le réseaux de neurones sur l'étendue des différents cas possibles.

Son objectif est de pouvoir rencontrer le maximum de cas distincts, afin de pouvoir récompenser les bonnes actions du CPU, sur tel et tel cas, lui permettant d'étoffer sa couverture des coups gagnants possibles.

Q] En laissant l'ordinateur jouer contre lui-même, que constatez-vous à la fin de N parties ? Expliquez la différence de scores entre les 2 joueurs.

Résultats de jeux :

En faisant jouer les deux joueurs CPU sur plusieurs sessions de 10000 parties, voici les différents résultats obtenus :

N°	Victoires du Joueur 1	Victoires du Joueur 2
1	5859	4141
2	6582	3418
3	9146	854
4	9011	989
5	9585	415
6	7214	2786
7	8935	1065

On constate que le joueur 1 dispose d'une position dominante vis-à-vis du joueur 2, ce qui implique que commencer la partie de jeu de bâtons est plus avantageux pour obtenir la victoire. Par ailleurs, en affichant les résultats par la méthode printAllConnections, on constate que le réseau de neurones du CPU1, possède des valeurs (connections neurales) bien plus élevées que celles du CPU2.

Interprétation :

Cependant, on peut constater que le ratio Victoires J2/Victoires J1 et soit très faible (ratio ≤ 0.10 dans la plupart des cas ci-dessus, soit plus élevé (0.3 ≤ ratio ≤ 0.4). Les valeurs intermédiaires apparaissent beaucoup moins fréquemment.

Cela implique que les premières parties doivent être déterminantes dans l'apprentissage, au sens où, les victoires permettent de forger un modèle plus robuste que le modèle adverse, et permet ainsi de remporter plus de parties.

C'est ce cercle vertueux, doublé de la position avantageuse du CPU1, qui explique son nombre de victoires écrasant vis-à-vis du CPU2 (cas 3,4,5,7). Cela explique également le nombre réduit de défaites du joueur 2 dans les cas 1, 2 et 6 : Le CPU2 a du gagner les premiers matchs, compensant ainsi sa position désavantageuse, par un apprentissage plus efficace.

Q] Refaire ces observations pour les autres modes

Résultats de jeux : En faisant jouer les deux joueurs CPU sur plusieurs sessions de 10000 parties, , avec cet fois-ci des difficultés différentes, voici les différents résultats obtenus :

Joueur 1/ Joueur 2	EASY	MEDIUM	HARD
EASY	4957/5053	562/9438	89/9911
MEDIUM	9469/531	5057/4943	802/9198
HARD	9383/617	9927/73	8139/1861

Explication :

Pour les IA facile et intermédiaire, leur comportement aléatoire est clairement mis en évidence par les résultats EASY/EASY et MEDIUM/MEDIUM, où le nombre de victoires est quasiment égal au nombre de défaites pour chaque joueur.

A noter que l'avantage de la gestion du dernier tour permet à l'IA intermédiaire de battre l'IA facile en terme de nombre de victoires totales.

Ce jeu de données confirme également les interprétations précédentes :

- La position du joueur 1 est plus avantageuse pour un joueur disposant d'une logique de jeu (cf HARD).
- Des victoires sur les premières parties permettent à l'IA difficile de devenir plus forte d'emblée : Cela se voit sur les matchs HARD/EASY, et HARD/MEDIUM, le second cas se termine avec moins de victoires pour le CPU1 que dans le premier cas, alors qu'il s'agissait d'un match plus simple à priori.
  Cela vient du fait que le modèle de l'IA difficile est aléatoire jusqu'à ce que le réseau de neurones soit correctement formé. L'IA facile a du remporter les premiers matchs retardant la croissance de l'IA difficile.

Jeu Final

Q] Jouez en mode "hard" en se plaçant "Joueur 2" (joueur qui ne commence pas). Est-ce possible de gagner ? Expliquer pourquoi.

Observations préalables :

- Aucune victoire n'a pu être réalisée contre l'IA en laissant la première place.
- Au premier tour, l'IA retire systématiquement 2 bâtons, laissant le joueur avec 13 bâtons.
- A chaque tour joué par l'utilisateur, le CPU s'arrange pour que le nombre de bâtons retiré sur les 2 derniers tours soient de 4 : Il jouera 3, quand vous jouez 1, etc.

Explication :

Il existe une méthode pour gagner à tous les coups au jeu du bâtons.

Pour cela, il faut faire en sorte que le nombre de bâtons au tour de l'adversaire s'exprime sous la forme 4n + 1 (n entier). Ensuite, on utilise le fait qu'en connaissant le nombre de bâtons de l'adversaire, on peut adapter son coup pour que le nombre de bâtons joués sur ces deux tours soit de 4 (1 + 3, 2 + 2, 3 + 1).

De cette manière, l'adversaire aura alors à jouer à son prochain tour, avec un nombre restant de 4(n – 1) + 1 bâtons. Ainsi, en itérant le processus (telle une propriété vérifiée par récurrence), l'adversaire se retrouvera à l'un de ses tours avec 4 * 0 + 1 = 1 bâton, le mettant irrémédiablement en position de défaite.

Au vu des observations, il s'agit de la technique que le CPU semble avoir assimilé : Il place le joueur dans ses conditions en le laissant avec 13 bâtons (= 4* 3 + 1), et agit de telle sorte à suivre le principe expliqué ci-dessus. Il est alors impossible pour le joueur de gagner face à l'IA dans ces conditions.