Puissance 4 - Monte Carlo Tree Search et Q-learning
Matière : Apprentissage Profond par Renforcement
Étudiants : Valentin PODDA, Froment Lorenzo

Lien vers le repository GitHub : (pour cloner le projet et suivre les indications du README.md)
https://github.com/FromentLorenzo/Puissance4MonteCarlo.git

Introduction :
Notre projet consiste à développer une intelligence artificielle pour le jeu de Puissance 4 en Python, en combinant les approches de Monte Carlo Tree Search (MCTS) et de Q-learning.

Le MCTS est particulièrement efficace pour évaluer les coups lorsque la partie est déjà avancée, car il peut anticiper les mouvements jusqu'à 10-15 coups à l'avance. Cependant, au début du jeu, MCTS peut manquer de direction et se retrouver dans des positions désavantageuses, car il n'a pas encore accumulé assez d'informations pour choisir les meilleurs mouvements.

C'est ici que le Q-learning devient un complément idéal. En permettant à l'IA d'apprendre des récompenses associées aux différentes positions et actions, le Q-learning aide l'algorithme à prendre des décisions plus éclairées dès le début de la partie. Cette synergie entre MCTS et Q-learning permet de construire une IA performante, capable d'optimiser sa stratégie à la fois dans les premiers coups et dans les phases avancées de jeu.

Analyse des Résultats
Le développement de notre agent hybride, combinant Monte Carlo Tree Search (MCTS) et Q-learning, montre des résultats prometteurs pour le jeu de Puissance 4. Nos tests comparatifs ont mis en évidence que l'agent hybride surpasse le MCTS pur dans les situations de jeu initiales et affiche des performances stables au fur et à mesure de l'avancement de la partie.

Q-learning vs MCTS

Temps par coup du MCTS (s)	Victoires du Q-learning	Egalité	Victoires du MCTS
0.01	70	1	29
0.05	62	0	38
0.1	53	1	46
0.5	42	1	57
1	33	0	67
2	24	1	75
5	19	0	81

Hybrid vs MCTS

Temps par coup du MCTS (s)	Victoires de l'hybride	Egalité	Victoires du MCTS
0.01	59	1	40
0.05	57	0	43
0.1	56	1	43
0.5	53	0	47
1	52	0	48
2	49	2	49
5	50	1	49

Performance du Q-learning seul vs MCTS
En comparant le Q-learning seul contre le MCTS, nous observons que le MCTS prend l'avantage à mesure que son temps de calcul par coup augmente. Lorsque le MCTS dispose de 2 secondes ou plus pour évaluer les coups, il dépasse nettement le Q-learning. En revanche, pour les temps plus restreints (0,01 à 0,5 seconde), le Q-learning gagne encore

un nombre significatif de parties. Cela confirme que le Q-learning permet de prendre des décisions solides rapidement, mais qu'il peine face à des calculs plus approfondis lorsqu'ils sont permis.

Performance de l'agent hybride vs MCTS

L'agent hybride (MCTS+Q-learning) montre des résultats équilibrés face à un MCTS utilisant un temps de calcul par coup prolongé. Lorsque le temps par coup du MCTS est supérieur à 1 seconde, les résultats deviennent très serrés, ce qui suggère que l'agent hybride conserve un niveau compétitif tout au long de la partie. Par exemple, avec 2 secondes par coup pour le MCTS, l'agent hybride et le MCTS pur terminent avec un score quasi égal (49 victoires chacun et 2 égalités), démontrant la capacité de notre agent hybride à rivaliser avec un MCTS même très optimisé en profondeur de recherche.

Pour les faibles temps par coup, l'agent hybride surpasse le MCTS, bénéficiant ainsi de l'apprentissage préalablement acquis via le Q-learning. Les décisions initiales de l'agent hybride sont alors mieux orientées, évitant les mouvements sous-optimaux qui peuvent se produire lorsque le MCTS manque de temps pour explorer suffisamment d'options.

Limites et Améliorations Potentielles

Bien que l'agent hybride montre des performances prometteuses, certaines limitations subsistent :

- **Équilibrage entre MCTS et Q-learning** : Actuellement, l'intégration des deux approches reste fixe. Une amélioration pourrait consister à adapter dynamiquement l'importance du Q-learning ou du MCTS selon l'avancement de la partie, le temps disponible, ou même le style de jeu de l'adversaire.
- **Exploration du Q-learning** : Le Q-learning peut être sujet à des problèmes de sur-apprentissage, particulièrement pour un jeu comme Puissance 4 où certaines configurations gagnantes ou perdantes peuvent être répétées. L'ajout d'une exploration continue ou d'une régularisation dans la Q-table pourrait encore renforcer les choix de l'agent hybride en début de partie.
- **Adaptation de l'algorithme de MCTS** : Un MCTS avec une fonction d'évaluation basée sur l'état de jeu (étudiée à partir des résultats du Q-learning) pourrait également accélérer le processus de décision et réduire le nombre de simulations nécessaires pour trouver un coup favorable.

Conclusion

En conclusion, l'association du Monte Carlo Tree Search et du Q-learning permet d'obtenir un agent performant et polyvalent pour le jeu de Puissance 4. Les deux approches se complètent en tirant parti de la capacité du Q-learning à explorer et stocker des informations utiles sur les premières configurations de jeu, tandis que le MCTS approfondit la recherche dans les phases finales. Cette combinaison, bien que simple, permet de réaliser un agent compétitif qui maintient une performance robuste quelle que soit la configuration de la partie.

En perspective, notre projet pourrait être étendu à d'autres jeux stratégiques ou même inclure des techniques de Deep Q-learning pour gérer des configurations de jeu encore plus complexes et faire de notre agent hybride un véritable modèle de prise de décision intelligente en environnement de jeu.

Sources :

Connect 4 Display using Pygame :
https://www.askpython.com/python/examples/connect-four-game
Connect 4 MCTS (Monte Carlo Tree Search) implementation :
https://www.harrycodes.com/blog/monte-carlo-tree-search#monte-carlo-tree-search-mcts