RATTRAPAGES DE RL 2024
Raphaël MOUROT

Introduction

Le Monte Carlo Tree Search (MCTS) est une méthode de recherche arborescente utilisée pour la prise de décision dans des environnements incertains, souvent utilisée dans les jeux de stratégie tels que les échecs. MCTS se base sur des simulations aléatoires pour explorer les différents états du jeu et est particulièrement efficace dans des domaines où l'espace de recherche est immense et les résultats déterministes (comme les échecs).

Les Quatre Étapes Principales du MCTS

1. Sélection
2. Expansion
3. Simulation
4. Rétropropagation

1. Sélection

La sélection consiste à parcourir l'arbre de décision depuis la racine jusqu'à une feuille en utilisant une politique de sélection, comme l'exploration et l'exploitation. L'objectif est de choisir les nœuds qui maximisent une certaine valeur de sélection. L'algorithme va la majorité du temps partir de l'état du jeu dont ils sait qu'il a le meilleur score, et regarder ses fils, c'est l'exploitation. Il va aussi aller dans un pourcentage plus faible des cas partir d'un état du jeu qu'il pense moins prometteur, mais qui pourrait évoluer mieux sur le long terme.

La sélection entre exploration et exploitation dans l'algorithme Monte Carlo Tree Search est une étape cruciale pour l'efficacité de l'algorithme. L'objectif est de choisir un enfant à chaque nœud de l'arbre de manière à maintenir un équilibre entre l'exploration de nouvelles possibilités et l'exploitation des choix qui semblent déjà prometteurs.

La formule UCT utilisée pour choisir le successeur iii dans un nœud de l'arbre est la suivante :

$$\frac{w}{n} + c\sqrt{\frac{\ln N}{n}}$$

où :

- w est le nombre de parties gagnées par le successeur i.
- n est le nombre de fois où le successeur i a été visité.
- N est le nombre de fois où le nœud parent de i a été visité.

- c est un paramètre d'exploration, souvent choisi expérimentalement, mais théoriquement égal à $\sqrt{2}$.

La formule se décompose en deux parties :

1. **Exploitation** w/n : Cette partie de la formule favorise les successeurs qui ont eu beaucoup de succès. Elle mesure le taux de victoires pour un successeur donné. Un successeur avec un taux de victoires élevé est préféré car il a montré de bonnes performances dans les simulations précédentes.

2. **Exploration** $(c\sqrt{\frac{\ln N}{n}})$: Cette partie favorise les successeurs qui ont été moins visités, encourageant ainsi la découverte de nouvelles stratégies potentielles. Elle est grande pour les successeurs qui n'ont pas été souvent explorés, incitant l'algorithme à explorer des choix moins connus qui pourraient s'avérer prometteurs.

Le paramètre c ajuste l'importance relative de l'exploration par rapport à l'exploitation. Un ccc élevé incite l'algorithme à explorer davantage, tandis qu'un ccc faible favorise l'exploitation des choix déjà connus pour être bons.

Les implémentations modernes de MCTS utilisent souvent des variantes de cette formule UCT pour améliorer l'efficacité de la sélection. Les travaux de Chang et al. ont notamment proposé des améliorations et des extensions à cette méthode, ce qui a permis d'affiner encore plus le compromis entre exploration et exploitation dans les algorithmes de décision multi-étage.

2. Expansion

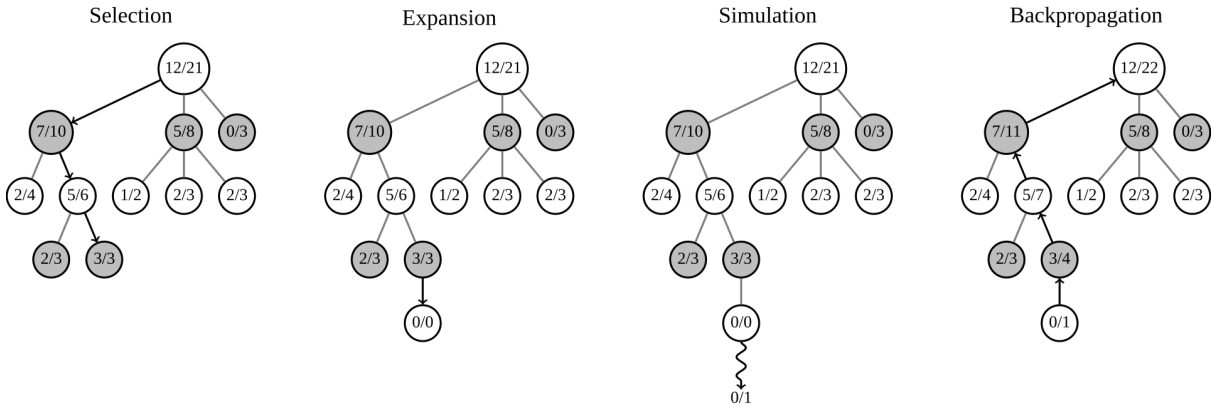
Lorsque la sélection atteint une feuille de l'arbre (un nœud sans enfant), si cet état n'est pas terminal (fin de la partie), de nouveaux nœuds sont créés en ajoutant tous les coups légaux possibles à partir de cet état.

3. Simulation

La simulation, aussi appelée "playout" ou "rollout", consiste à jouer aléatoirement ou semi-aléatoirement des parties complètes à partir des nœuds récemment ajoutés jusqu'à atteindre un état terminal. Le résultat de ces parties est utilisé pour estimer la valeur de l'état initial.

4. Rétropropagation

Après chaque simulation, les résultats sont propagés en remontant l'arbre, en mettant à jour les valeurs des nœuds visités durant la sélection avec les résultats des simulations. Ces valeurs peuvent inclure le nombre de victoires, de défaites, et de parties jouées, ce qui permet de réajuster les politiques de sélection pour les futurs parcours.



Sur cette illustration, on voit qu'on choisit le noeud fils 3/3 dans l'exploration, on va tester aléatoirement un coup, et ce coup va être estimé perdant, et on va donc augmenter le score "perdant" de tous les pères de ce noeud .

- Application aux Échecs :

Dans les échecs, l'expansion implique de prendre en compte tous les coups légaux pour une position donnée. Les simulations peuvent être guidées par des politiques simples (comme des heuristiques de base telles que manger toutes les pièces possible) ou par des réseaux de neurones entraînés pour évaluer les positions, ce que nous allons faire. On va utiliser un modèle entraîné pour copier stockfish, le meilleur algorithme des échecs, pour évaluer l'état du jeu actuel. Dans les phases d'ouverture et de milieu de jeu, les simulations peuvent être courtes et utiliser des bases de données ou des évaluations d'IA pour guider les coups. En fin de partie, des tables de fin de partie (endgame tablebases) peuvent être utilisées pour des
évaluations précises.

implémentation

1) Data processing

Pour télécharger les données, j'ai utilisé le site https://www.pgnmentor.com/files.html pour télécharger sous forme de PGN toutes les parties enregistrées que Kasparov à pu jouer, ainsi que des openings.

On va lire des parties d'échecs enregistrées dans des fichiers et utiliser un moteur d'échecs avancé, Stockfish, pour évaluer les positions dans ces parties. L'objectif est de créer un ensemble de données qui pourra être utilisé pour entraîner le modèle.

2) Entraînement du modèle

On va charger les données d'entraînement et de validation (utilisées pour vérifier la performance du modèle) à partir des fichiers contenant les parties d'échecs qu'on vient de processer dans la partie précédente. Les données sont organisées en lots pour permettre un traitement plus efficace.

Le but du modèle est de lui apprendre à raisonner comme Stockfish, et d'attribuer à chaque état du jeu d'échecs un score, selon si le coup est gagnant pour le joueur ou non.

3) Le MCTS

Ce programme utilise un algorithme basé sur Monte Carlo Tree Search (MCTS) pour décider quel coup jouer dans une partie d'échecs. Voici une description simplifiée du processus :

1. **Initialisation du Modèle** :

On charge le modèle de réseau de neurones qu'on à pré-entraîné dans la partie précédente.

2. **Définition des Nœuds de l'Arbre** :

Un arbre de recherche est créé, où chaque nœud représente une position d'échecs (un état du jeu).

Chaque nœud garde une trace des coups possibles dans ses nœuds enfants, des valeurs associées à ces coups, et du nombre de fois où chaque coup a été exploré.

3. **Sélection d'un Nœud Feuille** :

À partir de la racine (position actuelle), l'algorithme descend dans l'arbre en sélectionnant les coups les plus prometteurs jusqu'à atteindre un nœud feuille (une position qui n'a pas encore été entièrement explorée).

4. **Expansion du Nœud Feuille** :

Une fois qu'un nœud feuille est atteint, l'algorithme évalue cette position en utilisant le modèle de réseau de neurones pour obtenir des probabilités pour chaque coup possible et une estimation de la valeur de la position.

5. **Simulation** :

À partir de cette position, l'algorithme joue des coups au hasard jusqu'à ce que la partie se termine. Le résultat de la partie (victoire, défaite ou match nul) est utilisé pour mettre à jour les valeurs dans l'arbre de recherche.

6. **Propagation des Valeurs** :

Les résultats de la simulation sont ensuite propagés en remontant dans l'arbre, en ajustant les valeurs et les compteurs pour chaque coup joué lors de la descente vers le nœud feuille.

7. **Répétition du Processus** :

Ce processus (sélection, expansion, simulation, propagation) est répété de nombreuses fois pour affiner les estimations de valeur pour chaque coup possible à partir de la position actuelle.

8. **Sélection du Meilleur Coup** :

Après avoir effectué le nombre désiré de simulations, l'algorithme sélectionne le coup avec la valeur totale la plus élevée. Ce coup est considéré comme le meilleur coup à jouer dans la position actuelle.

En résumé, l'algorithme utilise une combinaison de recherche arborescente et d'évaluation par un réseau de neurones pour explorer différentes possibilités de coups et choisir le plus prometteur. Le processus d'exploration et d'évaluation est itératif et permet d'améliorer progressivement la précision des décisions prises par le programme.

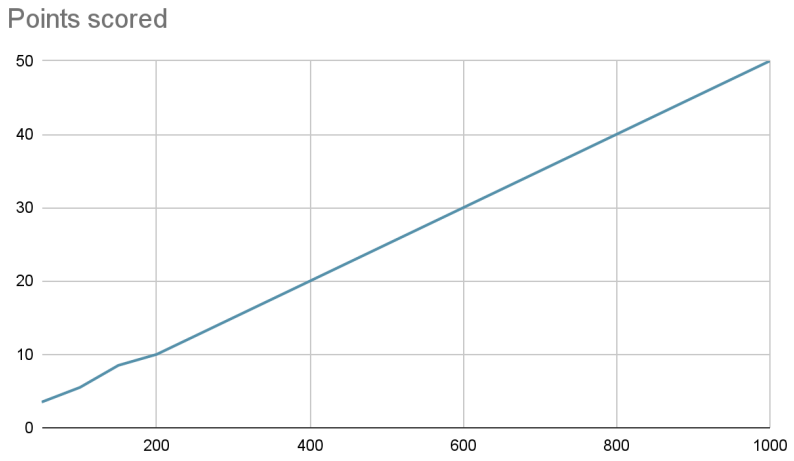
Résultats expérimentaux

MCTS	vs minimax	vs humain
modèle entrainé sur 150 epochs, 50 simulations	perd/boucle infinie	mat en ~5 tours
modèle entrainé sur 150 epochs, 200 simulations	perd/boucle infinie	mat en ~10 tours
modèle entrainé sur 1000 epochs, 50 simulations	perd/boucle infinie	mat en ~15 tours
modèle entrainé sur 1000 epochs, 200 simulations	gagne	mat en ~25 tours

J'ai testé 4 itérations du modèle, ou je varie le nombre d'epochs sur lesquelles a été entraîné le modèle, et le nombre de simulations par coup

Le MCTS a toujours tendance à commencer par faire bouger son cavalier en opening.
Quand il y a peu de simulations, il a tendance a bouger sa tour en aller-retours dans la case libérée par le cavalier ad eternam.
Il essaie de faire des mats du berger mais pas très bien, il oublie son fou.
On constate une nette amélioration dans le dernier cas lorsqu'on joue contre lui, il joue réactivement en défense de manière logique, mais manque d'attaque.

On a choisi 200 itérations car 10 sec est une vitesse de réponse acceptable, on constate que le temps pris par l'algorithme est linéaire, donc si on optimise le code, on pourrait aller plus vite.
pour référence, minimax met 30-35 sec



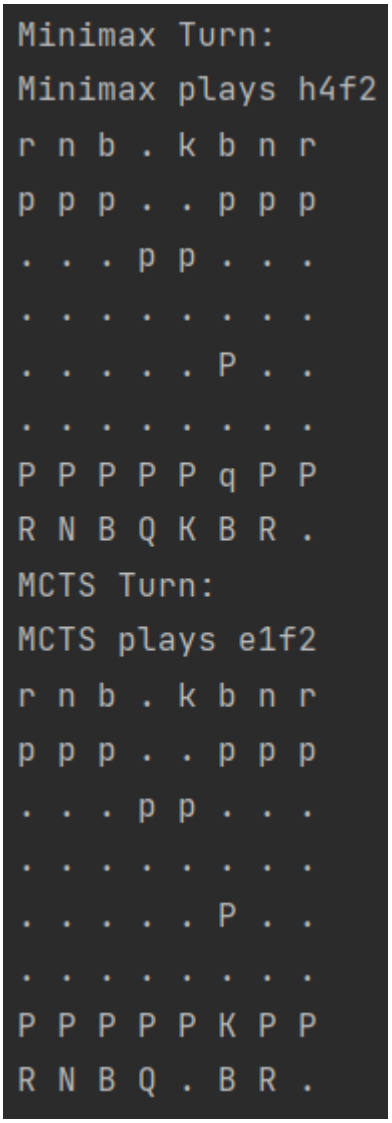


Illustration visuelle d’un exemple de terrible erreur de MCTS qui rate son mat du berger face à minimax

Conclusion.

On a donc compris la théorie du MCTS, puis implémenté ce dernier en suivant les 4 étapes, Sélection, Expansion , Simulation et Rétropropagation en l'adaptant aux échecs. Pour cela, on a implémenté un prétraitement de parties de Kasparov et de différents openings qu'on a ensuite utilisés pour entraîner un modèle pour évaluer les parties en essayant de reproduire les évaluations de Stockfish. On se sert de ce modèle dans MCTS pour évaluer la qualité d'un plateau durant la phase de simulation.

Ensuite, on a créé un environnement dans le terminal pour pouvoir le tester de manière visuelle. On l'a testé contre des humains et l'algorithme minimax avec différents hyperparamètres.

En pistes d'amélioration, on pourrait faire plus d'itérations de MCTS et d'epochs sur le modèle, mais ça nécessiterait plus de ressources que je n'ai pas.