Nina Lardiere
Tanguy Leclerc
Joran Marie

Synthèse

Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm

Introduction

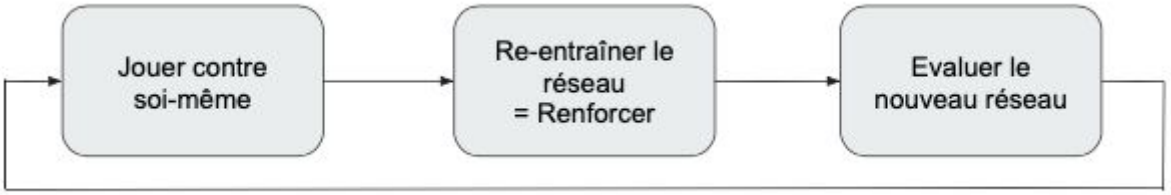
Le jeu des échecs et le jeu le plus étudié dans le domaine de l'histoire de l'intelligence artificielle. Babbage, Turing, Shannon et von Neumann ont créés du matériel, des algorithmes et des théories pour analyser et jouer aux échecs. Le but est évidemment de créer un joueur aux compétences surhumaines.
Le premier grand succès de l'informatique dans ce jeu fut Deep Blue contre Garry Kasparov le 11 mai 1997. Cette machine était entraînée avec un dataset d'anciennes parties, elle trouvait ses coups par parcours d'arbres avec des poids et fonctions d'évaluations créées à la main par de très bons joueurs d'échecs.
Avec la puissance grandissante des ordinateurs d'autres méthodes ont été proposées, comme des algorithmes de force brute. Ils consistent simplement à regarder toutes les combinaisons possibles et choisir une qui mène à la victoire. C'est une méthode très peu intéressante par sa nature. En 2002, l'ordinateur Hydra fut le premier à faire penser qu'aucun humain ne battrait plus une machine aux échecs.
Aujourd'hui deux approches se concurrences, Stockfish qui regarde et évalue 70 millions de positions par seconde et choisit son coup en fonction, et l'approche AlphaZero qui regarde 80 000 positions par secondes seulement mais qui s'appuie sur un réseau de neurones profond et entraîné par reinforcement learning.

Le problème abordé par l'article est donc qu'AlphaZero apprenne à jouer aux échecs par lui même, c'est à dire sans connaître rien d'autre que les règles du jeu. Aucune donnée préalable n'est donc utilisée par cette méthode.

Méthode

Ici, AlphaZero n'utilise pas de fonctions d'évaluation fabriquées par des experts et calculées par des heuristique, il utilise le Reinforcement Learning.

Rappel du principe général:



- **Jouer contre soi-même** correspond à **la création de données pour l'apprentissage**. Le meilleur réseau de neurones actuel va jouer 25 000 parties contre lui-même. A chaque mouvement, il enregistrera **l'Etat du jeu, les probabilités de mouvement** donnée par un algorithme de MonteCarlo et à la fin de la partie, **pour chaque mouvement, il attribuera +1 si le jour a gagné et -1 s'il a perdu.**

Note: Habituellement, les probabilités de mouvement sont donnés par un algorithme minimax effectuant une exploration complète de l'arbre de recherche. Ici, AlphaZero utilise la méthode de Monte-Carlo: chaque recherche correspond à une série de jeu simulé seul qui traverse l'arbre de la racine s_{root} jusqu'aux feuilles. Chaque simulation s'effectue en choisissant pour chaque état s , un mouvement a comportant les caractéristiques suivantes d'après le réseau de neurones actuel:

- encore peu visité
- avec une probabilité de mouvement élevée
- et une bonne issue de jeu z

Cette recherche retourne un vecteur π représentant la probabilité de distribution des mouvements, elle peut-être pris soit proportionnellement soit en privilégiant les actions peu visitées.

- **Renforcer le réseau** correspond à l'étape d'apprentissage du réseau. Le but est d'optimiser les poids à l'intérieur du réseau. Pour cela, on va effectuer 1000 boucles d'entraînements. Dans chaque boucle, on prend un mini-batchs de 2048 positions (Etat du jeu, probabilités de miuvements et +1/-1 en fonction de l'issue du jeu) issues des 500 000 dernières parties.

Le réseau de neurones est constitué d'une couche de convolution suivie de 40 couches résiduelles qui se séparent en deux têtes : le *value head* et le *policy head*. Ce réseau prend en entrée le plateau des positions noté s et ressort:

- un vecteur de probabilités de mouvement p avec $p(a) = P_r(a|s)$ pour chaque action a
- un scalaire v estimant l'issue de la partie z sachant les positions s : $v \approx E(z|s)$

Le but du réseau de neurones et d'**optimiser ses poids pour:**

- **Minimiser l'erreur** entre la valeur prédite v et le résultat du jeu déjà connu z.
  - La loss utilisée est l'**erreur quadratique moyenne.**
- Et aussi pour **maximiser la similarité entre le vecteur des probabilités prédites p et les probabilités de mouvements π** donnés en entrée par l'algo de MonteCarlo.
  - La loss utilisée est **l'entropie croisée.**

Et enfin, une fois les 1000 boucles effectués, on passe à la dernière section:

- **Evaluer le nouveau réseau:** On fait jouer l'ancien meilleur réseau de neurones et le nouveau sur 400 parties. Si le nouveau réseau de neurones gagne 55% des parties, il est nominé nouveau meilleur réseau, sinon il est tout simplement jeté.

Ainsi, le Reinforcement Learning est la répétition de ces 3 étapes encore et encore et encore…

Résultats

Afin d'évaluer l'algorithme AlphaZero, ses performances ont été comparées aux meilleurs programmes pour les jeux d'échecs (Stockfish), de shogi (Elmo), et de go (AlphaGo Lee et AlphaGoZero).

Une première évaluation a été faite pendant la phase "Jouer contre soi-même" de reinforcement learning, basée sur le classement Elo. Le classement Elo est un système d'évaluation comparatif du niveau de jeu des joueurs de jeux en un contre un. Un nombre de points Elo est attribué au joueur, suivant ses performances passées, tel que deux joueurs supposés de même force aient le même nombre de points.

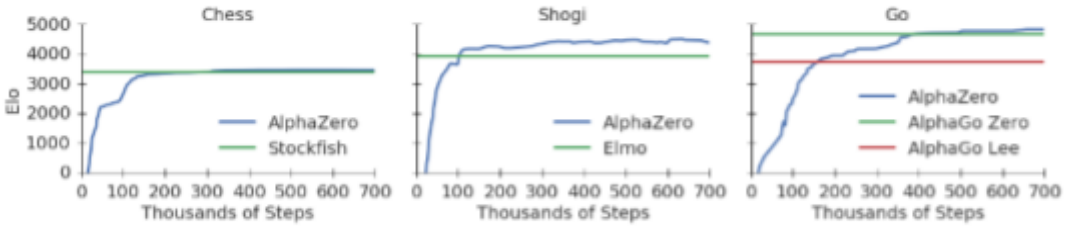


Figure 1: TrainingAlphaZerofor 700,000 steps. Elo ratings were computed from evaluation games between different players when given one second per move.

On peut voir sur la Figure 1 qu'aux échecs AlphaZero a été plus performant au bout de 300 steps, au shogi au bout de 110 steps et au go au bout de 165 et 400 steps.

Une autre manière d'évaluer la méthode est de faire jouer le programme contre les meilleurs de chaque jeu. Ainsi pour un tournoi de 100 matchs, AlphaZero enregistre 0 défaite aux échecs, 8 au shogi et gagne 60 fois au jeu de go. Alpha Zero bat donc tous ses concurrents de manière convaincante.
Enfin la performance du MCTS d'AlphaZero à aussi été comparée aux recherche alpha-beta de Stockfish et Elmo. Si AlphaZero recherche moins de positions par seconde il compense en se concentrant sur les variations les plus prometteuses. On remarque que le MCTS d'AlphaZero s'adapte mieux au temps de réflexion remettant en cause la supériorité des méthodes alpha-beta dans ce domaine. Enfin l'approche "plus humaine" du MCTS permet à l'algorithme de découvrir et d'utiliser les ouvertures de jeu d'échecs les plus couramment utilisées par les humains, ce qui prouve qu'il maîtrise un large spectre du jeu.

Conclusion

Grâce au Reinforcement Learning on arrive très vite à des modèles plus performants mais ce n'est pas leur meilleure qualité. Ces modèles sont très génériques, il suffit juste de

coder un autre jeu et l'entraîner pour arriver à des performances imbattables. Leur approche "humaine" des jeux est aussi un grand plus et apporte une élégance, chose très importante pour les amateurs d'échecs et même d'informatique. En effet lorsque AlphaZero décide d'un coup à jouer , même si aucun humain n'y avait pensé, ce coup fait sens et les experts sont capables de le comprendre. Ce qui n'est pas le cas des autres modèles qui ont une approche bien plus "long terme" des parties et qui joue pour de petits gains jusqu'à une victoire lente.
Aujourd'hui AlphaZero est la meilleure intelligence informatique d'échecs au monde et est utilisé par les meilleurs joueurs d'échecs pour trouver de nouveaux coups et s'entraîner. Cette association élégance et humanité nous paraissent très intéressantes comparés à une approche force brute.

Source

Mastering Chess and Shogi by Self-Play with aGeneral Reinforcement Learning Algorithm : https://arxiv.org/pdf/1712.01815.pdf