

Deep Checker

Projet de statistiques ENS Rennes 2020

Arthur Correnson Igor Martayan Manon Sourisseau

Introduction

Dans ce projet, nous nous intéressons au jeu de dame. L'objectif est d'utiliser des méthodes d'apprentissage statistiques pour construire une intelligence artificielle capable de jouer au jeu. L'idée maîtresse étant d'apprendre une heuristique évaluant la qualité des coups. Pour se faire, nous utiliserons des données récoltées lors de la simulation d'un grand nombre de parties de jeu de dame. Notons que ce projet est un projet de bout en bout, c'est à dire qu'il couvre [1] la génération du jeu de donnée, [2] le choix des modèles et enfin [3] l'entraînement des modèles et leur mise en pratique.

1 - Génération des données & création d'un simulateur de jeu de dames

Les techniques d'apprentissage statistiques utilisent, comme leur nom l'indique, des données comme matière première. Pour avoir les données nécessaires à l'apprentissage d'une bonne heuristique, nous avons commencé par créer notre propre simulateur de jeu de dames. Pour un bon apprentissage, il est primordial de disposer d'un grand nombre de données, sous peine d'obtenir une heuristique qui serait sur-spécialisée pour les quelques scénarios présentés par le jeu de données mais qui se généraliserait très mal aux autres scénarios de partie possibles. Afin de générer un grand volume de données, il était donc nécessaire d'avoir un simulateur qui offre des grandes performances, mais également une représentation très compacte des données pour éviter de faire exploser la mémoire nécessaire à la simulation des parties et aux stockages des informations collectées. Nous avons donc fait le choix d'écrire le simulateur en langage C et de définir par la même une représentation binaire astucieuse des données de jeu.

Sans rentrer dans les détails précis de l'implémentation du simulateur, en voici une présentation générale. Nous commençons par constater qu'un damier possède 64 cases dont seul 32 peuvent-être occupées par des pions. Par ailleurs, une case est soit vide, occupée par un pion blanc ou occupée par un pions noir. On peut donc modéliser un état du damier comme deux entier 32 bits (que l'on combinera en un unique entier 64 bits) chacun représentant lesquelles des 32

cases sont occupées par les pions noirs (respectivement blancs). La simulation d'une partie complète est ensuite réduite au calcul d'opération bit à bit sur l'état du damier.

Le simulateur garde une trace complète de chaque partie sous la forme d'une liste des états pris par le plateau. La séquence de chaque partie simulée est ensuite écrite dans un fichier texte permettant ainsi le traitement des données à posteriori.

La version finale du simulateur offre des performances remarquables. Plusieurs milliers de parties peuvent-être simulées en moins d'une seconde sur un ordinateur classique. Notons que l'essentiel du temps de simulation est consacré à l'écriture des données dans un fichier texte. Une amélioration possible serait donc d'écrire les données binaires directement dans un fichier afin d'éviter les temps d'entrées sorties trop long et les surcoût lié à la conversion des données binaires vers des données textuelles. Nous avons tout de même fait le choix de conserver une représentation finale textuelle afin de faciliter la lecture des données par des outils externes (tableurs, scripts python, etc).

2 - Modèles et heuristiques

Rappelons notre objectif est de construire une fonction (heuristique) permettant d'attribuer à un coup quelconque un score représentant intuitivement sa "qualité". Par souci de simplicité dans les modélisations, on s'intéressera uniquement aux coups de l'un des deux joueurs que l'on appellera joueur 1 (l'autre sera appelé joueur 2). A symétrie du damier près, les résultats obtenus pour noter les coups du joueur 1 sont toujours généralisables aux coups du joueur 2. Nous présentons plusieurs approches pour déterminer une *bonne* heuristique.

Un premier système de notation des coups

Soit \mathcal{DB} une collection de parties simulées et soit P_i un élément de \mathcal{DB} . P_i est une suite de damiers de laquelle on peut extraire l'ensemble C des coups c joués au cours de la partie. Un coup c pourra simplement être vu comme une paire de damier représentant l'état du jeu avant et après le coup. Notons qu'au jeu de dame, un même coup ne peut se produire qu'une seule fois par partie au plus. Cette singularité est due au système de règles qui interdit de jouer en arrière.

Modélisation du problème Si X est l'univers de tous les coups possibles du joueur 1 au jeu de dame, nous cherchons maintenant à trouver une fonction $h : X \rightarrow [-1, 1]$ où $h(x) = 1$ si x est un excellent coup, -1 si c'est un très mauvais coup. Un premier squelette partiel de la fonction h peut-être construit à partir de nos données de simulation \mathcal{DB} comme suit :

- Pour chaque partie $P_i \in \mathcal{DB}$ on calcul C_i l'ensemble des coups joués par le joueur 1 au cours de P_i

- On équipe chaque P_i d'une *distance* notée $\|\cdot\|_i : C_i \rightarrow \mathbb{N}$ qui caractérise la distance à la victoire d'un coup dans P_i et définie comme : $\|c\|_i = d(c).v(c)$ où $d(c) \in \mathbb{N}$ est le nombre de coups qui séparent c de la fin de partie et $v(c) = 1$ si le joueur 1 a gagné et $v(c) = 0$ sinon
- On construit maintenant une fonction $w_i(c)$ telle que $w_i(c) = \|c\|_i$ si $c \in C_i$ et $w_i(c) = 0$ sinon
- Pour chaque coup c apparaissant dans l'ensemble des parties de \mathcal{DB} , $h(c) = \frac{1}{N} \sum w_i(c)$ avec N le nombre de i tel que $w_i(c) \neq 0$

Pseudo-apprentissage par régression aux k plus proches voisins Une fois le squelette de h élaboré, on souhaite le généraliser afin de pouvoir noter un coup c quelconque n'apparaissant pas forcément dans \mathcal{DB} . Pour se faire, on utilise une méthode dite de régression aux k plus proches voisins (KNN). Cette méthode consiste essentiellement à généraliser le squelette de h de sorte que pour tout coup c hors du domaine de h , on attribue à c le score moyen des k coups les *plus proches* de c dans le domaine. La notion de *plus proches* s'obtient naturellement à partir de la définition d'une distance. Nous choisissons ici d'introduire la distance entre coups $\langle \cdot, \cdot \rangle_{KNN} : X \rightarrow \mathbb{N}$ définie comme la distance d'édition entre deux damiers. Intuitivement, il s'agit de compter le nombre de pions qui diffèrent entre les deux coups vu comme des couples de damiers.

Remarque Dans notre implémentation, les coups sont représentés sous forme de séquences de 128 bits. Le calcul de la distance entre deux coups se réduit alors à un calcul de XOR bit à bit suivi du comptage du nombre de bits à 1 :

$$\langle c_1, c_2 \rangle_{KNN} = \|c_1 \oplus c_2\|_1$$

On notera $KNN(k, c)$ le score obtenu pour le coup c par régression aux k plus proche voisins.

Mise en pratique de KNN Dans une situation de plateau donnée, notre heuristique permet calculer le coups suivant à jouer de la manière suivante :

On commence par déterminer l'ensemble des coups c_i jouables. Le prochain coup à jouer est alors $\arg \min_{c_i} KNN(k, c_i)$

Empiriquement, nous avons choisi une valeur de $k = 4$. Ce choix peut être sujet à discussion.

Conclusion