Rapport DataScience & IA : Tic Tac Toe

Structure de données :

Notre morpion se joue dans une grille 12x12, correspondant dans notre code à une liste de liste contenant initialement des points en chaque position. A chaque fois qu’un joueur joue, la position demandée est remplacée par un X ou un O.

Afin de définir une action, nous avons créé une classe Action composée de deux attributs :

* Un tuple correspondant aux coordonnées
* Un caractère correspondant à son motif

Pour sauvegarder l'ensemble des actions disponibles au sein d'une grille nous avons créé la fonction action qui parcourt la grille et sauvegarde dans une liste l'ensemble des coordonnées i, j ayant pour motif un point, sous la forme de tuples. De plus pour insérer le coût d'un adversaire au sein de la grille nous avons créé la fonction Result qui prend la grille en paramètre ainsi qu'une action composée d’un motif et des coordonnées. Cette fonction retourne la grille mise à jour.

TerminalTest(grille) :

Afin de vérifier si une partie est terminée nous avons aussi créé la fonction terminal test qui vérifie sur l'ensemble des lignes, des colonnes et des diagonales si 4 pions sont alignés à la suite. La fonction appelle donc trois sous fonctions s’appelant CheckLine, CheckColumn et CheckDiag qui sont des sous fonctions permettant la vérification des alignements.

MyHeuristiques(grille) :

L’heuristique de notre code a sans doute été la partie la plus complexe étant donné qu'elle représente le centre de la réflexion de notre intelligence artificielle. En effet c'est cette fonction qui va permettre la notation de chaque état de la grille au cours du jeu en fonction de sa profondeur. L’Heuristique va suivre les différentes règles que nous lui imposons, afin d’accorder plus ou moins de points à l’état de la grille. Voici les règles pour les lignes, les colonnes et les diagonales :

* 3 pions alignés et deux cases libres aux extrémités -> +100
* 2 pions alignés et deux cases libres aux extrémités -> +10
* 2 pions un espace et un autre pion -> +30
* 1 pion un espace et 2 autres pions -> +30

Nous parcourons la grille et nous ajoutons donc à une variable scoreX les règles que les pions X vérifient, de même pour les pions O dans une variable scoreO, puis nous retournons la différence qui correspond à l’évaluation de la grille. Cette fonction a été la plus complexe étant donné qu'il était possible de mettre toutes les règles que nous voulions, le but étant d’obtenir une évaluation la plus efficace possible, et c’est ce schéma que nous avons trouvé comme le meilleur.

IA(grille) :

La fonction qui fait jouer un coup à notre ordinateur, prend donc la grille en paramètres, et va ensuite boucler sur l’ensemble des actions possibles en récupérant le résultat de notre fonction MinMax\_Decision sur chacune d’elles. Si le résultat est meilleur que le précédent, nous le sauvegardons comme meilleur score, ainsi que les coordonnées de la position du coup dans la grille. A la fin de la boucle, nous ajoutons le meilleur coup retenu dans la grille et nous renvoyons la grille pour que le jeu se poursuive.