AgroParisTech UC 2A « La place des bandits en IA »

TP « L'algorithme MinMax »

https://antoinecornuejols.github.io/teaching/AGRO/UC-2A-Bandits/UC-2A_IA_Bandits.html

---

1. Introduction

Minimax est un algorithme de prise de décision pour les jeux à deux joueurs (comme les échecs, le tic-tac-toe ou les dames). Il permet au joueur, appelons-le J_1 (i.e. l'ordinateur) de choisir le coup à jouer dans la situation courante tout en supposant que son adversaire fera toujours le meilleur coup possible pour lui-même, J_2, c'est-à-dire le plus défavorable pour J_1. Dans cette section, nous allons examiner un aperçu de l'algorithme à travers l'exemple du tic-tac-toe.

La fonction MinMax prend trois arguments en entrée :

1. La position courante du jeu
2. La profondeur à laquelle elle peut explorer (0 si c'est une feuille de l'arbre de recherche)
3. Si c'est au tour du joueur Max de jouer ou au tour du joueur Min.

2. L'algorithme MinMax

L'algorithme MinMax suppose qu'il y a deux joueurs en compétition. Le joueur J_1 ou joueur `Max` à qui c'est le tour de jouer et le joueur J_2 ou joueur `Min`, son adversaire.

Quand c'est à un joueur de jouer et de choisir un coup, il endosse le rôle de joueur `Max` puisqu'il essaie de maximiser son gain.

Étant données ses ressources de calcul et de mémoire, il détermine la profondeur maximale $d_{\max}$ à laquelle il peut explorer l'arbre des possibilités. Cette profondeur dépend bien sûr du *facteur de branchement* du jeu. Moins de 9 pour le Tic-Tac-Toe à 9 cases, de l'ordre de 30 en moyenne pour le jeu d'échecs et partant de 357 pour le jeu de Go, en diminuant à chaque nouveau pion déposé.

L'exploration consiste alors à développer l'arbre des possibilités jusqu'à cette profondeur limite ou jusqu'à ce qu'une condition d'arrêt du jeu soit rencontrée : gain ou perte du joueur ou partie nulle. On appelle les situations correspondantes *feuilles* de l'arbre.

Chaque feuille est alors évaluée par une *fonction d'évaluation*. Celle-ci évalue la position correspondante en fonction du joueur `Max`.

- Si les feuilles correspondent à des fins de partie, la fonction d'évaluation retourne une valeur pour le gain de la partie, par exemple : 20, la perte serait alors évaluée à -20, et une partie nulle 0.
- Si la feuille ne correspond pas à une fin de partie, la valeur de la position est évaluée en fonction de sa qualité pour le joueur `Max`.

Les valeurs de ces feuilles sont alors remontées par les fonctions Min et Max jusqu'à la racine de l'arbre : le joueur `Max` qui choisit la branche correspondant à la valeur la plus élevée.

La qualité de la fonction d'évaluation est évidemment déterminante quand l'arbre n'atteint pas les situations terminales de jeu pour lesquelles on sait quelle est l'issue de la partie. C'est le rôle de la fonction `eval` dans le pseudocode ci-dessous.

```
min_max(position, profondeur, joueur) =
  si situation est finale ou profondeur = 0  (feuille de l'arbre)
      retourner eval(situation, joueur) x coeff(joueur)  (+1 si joueur Max, -1 sinon)
  sinon
      Liste_pos_suivantes = les positions suivantes (légales) de position
      si joueur = 'Max'
         valeur = max( [ min_max(suivante, profondeur-1, 'Min')
                       pour suivante in Liste_pos_suivantes ] )
       sinon # joueur = 'Min'
         valeur = min( [ min_max(suivante, profondeur -1, 'Max')
                        pour suivante in Liste_pos_suivantes ] )
      fin si
      retourner valeur
  fin si
```

Dans la suite de ce TP, nous allons d'abord étudier un programme pour jouer à Tic-Tac-Toe (3 $\times$ 3) avec une exploration jusqu'aux situations terminales.

Puis nous l'adapterons pour jouer au jeu de dans lequel nous introduirons une fonction d'évaluation pour des feuilles non terminales.

3. L'algorithme MinMax pour le jeu de Tic-Tac-Toe

Vous trouverez le code correspondant à un algorithme MinMax de jeu pour Tic-Tac-Toe dans le fichier `TicTacToe.py`.

Vous le lancez par la commande : `python TicTacToe.py`.

Vous pouvez jouer quelques parties en tant que joueur humain.

Examinez le code pour en comprendre les différentes fonctions.

4. L'algorithme MinMax pour le jeu de Dodgem

Dans un premier temps, vous allez modifier le programme pour qu'il joue au jeu de Dodgem. Voir `https://gamescrafters.berkeley.edu/games.php?game=dodgem` pour les règles.

Dans un second temps, vous allez coder une fonction d'évaluation qui puisse se déclencher avant une situation terminale.

Et vous expérimenterez.