Rapport DIA Projet Ultimate Tic-Tac-Toe

A3S6 TD-D

Hugo CHEN, Nathan CHOUKROUN, Enzo DELGADO, Raphaël DEMARE

Tout d'abord, nous avons décidé de choisir une base en string pour le tableau de 9*9 grâce à la fonction index(x,y) comme le tableau à droite.

0	1	2	9	10	11	18	19	20
3	4	5	12	13	14	21	22	23
6	7	8	15	16	17	24	25	26
27	28	29	36	37	38	45	46	47
30	31	32	39	40	41	48	49	50
33	34	35	42	43	44	51	52	53
54	55	56	63	64	65	72	73	74
57	58	59	66	67	68	75	76	77
60	61	62	69	70	71	78	79	80

Parce que nous pensons qu'une base en listes pouvait être gênant à cause des numérotations différentes (-1 en indices de array ou liste) et aussi que le jeu Ultimate Tic-Tac-Toe utilise des caractères "X", "O" et ".".

Le but d'IA est de trouver les meilleurs endroits à jouer pour qu'il puisse gagner. Ici, nous devons seulement utiliser l'algorithme de minimax avec alpha beta. Comme nous avons déjà utilisé cet algorithme pendant les cours, il nous reste simplement à adapter le minimax à l'échelle de l'Ultimate Tic-Tac-Toe.

Dans un premier temps, nous allons obtenir les coups possibles pour le joueur actuel. Nous sélectionnons la petite case à jouer en prenant le reste de la division d'indice du coup précédant par 9. Puis nous vérifions si la case est déjà occupée. Si elle l'est, alors nous prendrons toutes les petites cases qui peuvent être jouer. Sinon, nous prendrons cette petite case. Pour pouvoir faire des prévisions, nous enregistrons tous les endroits vides parmi les coups possibles et les tableaux du prochain coup.

Grâce aux prévisions, nous pouvons commencer à construire notre propre minimax. Nous savons que l'Ultimate Tic-Tac-Toe possède des règles spécifiques et complexes pour jouer, comparé au Tic-Tac-Toe de 3*3. Alors nous devons limiter la profondeur de calcul et la hauteur de l'arbre de recherche, ce qui n'est pas le cas pour un Tic-Tac-Toe normal. Par la suite, avec un temps de calcul limité, nous avons choisi la profondeur = 6 (depth) pour avoir le temps de développer tout l'arbre de recherche et avoir plus de performance.

L'algorithme du minimax va choisir la meilleure branche de l'arbre de recherche en essayant tous les coups et les tableaux possibles à la suite. Il réalise cette étape en maximisant les coups du joueur actuel et minimisant les coups de l'adversaire. Comment l'algorithme peut évaluer la situation des tableaux atteints après le placement de chaque coup ?

Pour ce faire, nous devons construire une évaluation en points sur les tableaux qui est fondamental pour notre code. Le but est d'attirer l'IA à gagner plus de points possibles pour nous rapporter la victoire.

Nous listons d'abord les objectifs possibles pour un Tic-Tac-Toe, c'est de gagner en rangée, en colonne ou en diagonale. Nous utilisons une fonction "Counter()" de la librairie "collections" pour compter le nombre d'occupations par les joueurs. Donc pour une case de 3*3, nous utilisons la pondération sur la notation de points. Nous rajoutons des points différents pour les pièces de l'IA selon le nombre d'occupations qui sont alignés sur les objectifs possibles dans la case. Nous enlevons aussi des points si ce sont les pièces de l'adversaire. Nous écartons les coefficients donnés aux différentes situations pour éviter que l'IA préfère des mauvais coups. Comme d'avoir plusieurs pièces alignées ou la victoire d'une case. Si c'est une case nulle ou une ligne nulle, alors nous enlevons également des points de coefficient, car nous privilégions la victoire. Celle-ci peut aussi être utilisée sur une liste qui enregistre la situation de 9 petites cases. Nous donnons alors un coefficient dense sur cette évaluation qui détermine directement le résultat du jeu. Nous avons également ajouté des points bonus pour la profondeur, c'est à dire, l'IA envisage de gagner avec moins de coups possibles.

Les fonctions de maximisation et minimisation s'arrêtent alors quand la profondeur est arrivée à 0 ou qu'un joueur gagne ou le tableau est plein. A ce moment, elles renvoient l'évaluation du score sur les nouveaux tableaux.

Nous prenons ensuite des valeurs d'entrées valides selon les coordonnées de chaque coup. Nous laissons la possibilité au joueur de choisir qui commence le jeu et le choix de la pièce.

Nous utilisons seulement l'algorithme Minimax du Machine Learning, mais sa performance pourrait être meilleure. Ce projet nous encourage à continuer d'étudier les différents algorithmes de data science et d'IA. Nous avons apprécié pouvoir construire nous-mêmes ce projet intéressant.