IA *Découverte de l'intelligence artificielle*

TD no5	**IA pour jouer à 2048**

Objectif de la séance :
Développez une IA utilisant le méthode de Monté Carlo

1 Avant propos

1.1 En python, les fonctions sont des objets comme les autres !

En python les fonctions sont des objets comme les autres. L'identifiant d'une fonction référence un objet de type fonction. Ainsi la déclaration suivante définie une variable `foo` qui referénce un objet de type fonction :

```
def foo():
    print("foo")
```

On peut donc affecter une fonction à une nouvelle variable et exécuter cette fonction à l'aide de cette nouvelle variable :

```
>>> foo_bis = foo
>>> foo_bis()
foo
```

De plus, une fonction peut alors être passée en paramètre d'une autre fonction. Par exemple, le code suivant déclare une deuxième fonction `bar` et une fonction `f` qui prend en paramètre une fonction et qui l'exécute :

```
def bar():
    print("bar")

def f(une_fonction_sans_parametre):
    une_fonction_sans_parametre()
```

On peut donc utiliser la fonction `f` avec comme paramètre effectif la fonction `foo` ou la fonction `bar` :

```
>>> f(foo)
foo
>>> f(bar)
bar
```

Cette possibilité est par exemple utilisée pour bien séparer la logique métier de l'interface homme machine (IHM). Nous l'avons utilisée dans les premiers TP sur le Sudoku et le Puissance 4. Par exemple la fonction `jouer` du Puissance 4 a la signature suivante :

```
def jouer_une_partie(obtenir_coup_jaune: "Fonction(Plateau,
    Couleur) -> int",
                     obtenir_coup_rouge: "Fonction(Plateau,
                        Couleur) -> int") \
                     -> "PARTIE_NULLE|JAUNE_GAGNE|ROUGE_GAGNE":
    """ Fonction qui permet de jouer une partie de
    puissance 4 en donnant deux fonctions qui permettent
    d'obtenir, à chaque tour de jeu, un coup pour les jaunes
    et pour les rouges"""
```

Ainsi en fonction des paramètres formels que l'on utilise réellement, les coups des jaunes et des rouges peuvent être obtenus depuis la saisie d'un joueur humain ou grâce à une IA.

1.2 La méthode de Monté Carlo

Pour certains problèmes, la recherche exploratoire pour identifier la meilleure solution n'est pas imaginable car le nombre de solutions est astronomique [1] voire crytographique [2].

« La méthode Monté Carlo désigne une famille de méthodes algorithmiques visant à calculer une valeur numérique approchée en utilisant des procédés aléatoires, c'est-à-dire des techniques probabilistes. » (Wikipédia).

Elle est par exemple utilisée dans les jeux dont la taille de l'espace des solutions est cryptographique (jeu d'échecs [3], jeu de go [4], etc). Son principe est le suivant : soit une position P avec n coups possibles. Pour chaque coup on simule N parties entièrement jouées aléatoirement. Le coup retenu est celui dont une heuristique (nombre de parties gagnées, moyenne/max des scores des parties, etc.) aura la plus grande valeur.

1.3 Le jeu 2048

D'après Wikipédia, « 2048 est un jeu vidéo de type puzzle conçu en mars 2014 par le développeur indépendant italien Gabriele Cirulli et publié en ligne sous licence libre via Github le 9 mars 2014.

Le but du jeu est de faire glisser des tuiles sur une grille, pour combiner les tuiles de mêmes valeurs et créer ainsi une tuile portant le nombre 2048. »

La figure 1 présente une copie d'écran de ce jeu.

2 Une IA pour le jeu 2048

Récupérez et décompressez l'archive `2048.zip` disponible sur Moodle. Cette archive est composée des modules python suivants :

Module grille_2048.py qui propose des fonctions permettant de manipuler une grille de 4x4 du jeu 2048. La case se trouvant en haut à gauche a les coordonnées $(1, 1)$. Les fonctions de ce module sont :

1. Un nombre est dit astronomique lorsqu'il représente l'univers (nombre de jours depuis la naissance de l'univers, nombre de particules dans l'univers, etc.). Il est de l'ordre de 10^n avec n qui vaut quelques dizaines.
2. Un nombre est dit cryptographique lorsqu'il est utilisé pour chiffrer en toute sécurité des messages. Il est de l'ordre de 10^n avec n qui vaut plusieurs centaines.
3. Le nombre de Shannon, soit 10^{120}, est une estimation de la complexité du jeu d'échecs, c'est-à-dire du nombre de parties différentes, ayant un sens échiquéen, possibles (Wikipédia)
4. Taille estimée à 10^{700}, cf. `https://go.quebecjeux.org/wp-content/uploads/2019/01/Nombres.pdf`

FIGURE 1 – Une copie d'écran du jeu 2048

grille_vide qui permet d'obtenir une grille vide ;

case_vide qui permet de savoir si une case est vide ;

obtenir_valeur qui permet d'obtenir la valeur d'une case non vide ;

fixer_valeur qui permet de modifier la valeur d'une case ;

vider_case qui permet de vider une case ;

remplie qui permet de savoir si une grille est totalement remplie ;

obtenir_case_vide qui permet d'obtenir aléatoirement les coordonnées d'une case vide ;

grille_en_chaine qui permet d'obtenir la représentation en chaine de caractères d'une grille ;

copier qui permet d'obtenir une copie d'une grille.

Module jeu_2048.py qui propose des fonctions de logique métier pour jouer au jeu 2048. Ce module propose quatre constantes qui correspondent aux quatre directions (`VERS_HAUT`, `VERS_BAS`, `VERS_GAUCHE`, `VERS_DROITE`). Les fonctions de ce module sont :

ajouter_une_valeur qui permet d'ajouter une valeur aléatoire dans une case vide d'une grille ;

deplacer qui permet de déplacer les valeurs des cases dans une direction en les fusionnant suivant la règle du jeu. La valeur retournée, que l'on nomme gain, est la somme des nouvelles valeurs créées par ces fusions ;

directions_possibles qui permet d'obtenir les directions possibles ;

grille_de_depart qui permet d'obtenir une grille initialisée avec deux 2 positionnés au hasard ;

jouer qui permet de jouer à 2048 à partir d'une grille donnnée. Cette fonction retourne le score en fin de partie et le nombre de coups joués ;

faire_une_partie qui permet de faire une partie de 2048. Cette fonction retourne le score en fin de partie et le nombre de coups joués ;

Module jeu_2048_ihm_texte.py un script qui permet à un humain de jouer à 2048 en mode texte. Ce script contient les fonctions suivantes :

afficher_grille qui affiche une grille sur la sortie standard ;

obtenir_direction qui demande à l'utilisateur de saisir sur l'entrée standard une direction (en utilisant les quatre chiffres du pavé numérique associés aux quatre directions : 4 pour gauche, 8 pour haut, 6 pour droite et 2 pour bas) ;

main le programme principal.

Module jeu_ia_2048_monte_carlo.py un script qui permet à une IA de jouer à 2048 à l'aide de la méthode de Monté Carlo. Au lancement du script, l'utilisateur peut configurer le nombre de parties jouées pour chaque coup possible (par défaut 10). Ce script contient les fonctions suivantes :

obtenir_direction_aleatoire qui retourne une direction aléatoire ;

obtenir_direction qui retourne une direction déterminée par la méthode de Monté Carlo avec comme calcul de l'heuristique la moyenne des scores des parties jouées aléatoirement ;

main le programme principal.

2.1 Logique métier

Développez la fonction `jouer` du module `jeu_2048` sachant qu'une partie est en cours tant que le joueur peut choisir des directions ou qu'il n'a pas atteint 2048 (le gain lié au déplacement est supérieur ou égal à 2048). L'algorithme informel de cette fonction est :

- initialiser des variables locales (nombre de coups, partie terminée ?, directions possibles) ;
- tant qu'il y a des directions possibles et que la partie n'est pas terminée, il faut :
  - afficher la grille ;
  - obtenir la direction choisie (en utilisant le paramètre formel) ;
  - réaliser le déplacement correspondant et obtenir le gain ;
  - incrémenter le nombre de coups ;
  - incrémenter le score avec le gain ;
  - ajouter une valeur aléatoire ;
  - obtenir les nouvelles directions possibles ;
  - indiquer que la partie est terminée lorsque le gain est supérieur à 2048.
- afficher la grille ;
- retourner le score final et le nombre de coups.

Testez votre fonction en jouant à 2048 à l'aide du script `jeu_2048_ihm_texte`.

2.2 L'IA

Développez la fonction `obtenir_direction` du script `jeu_2048_ia_monte_carlo` qui affichera pour chaque coup joué les informations justifiant le choix. L'algorithme informel de cette fonction est :

- déterminer les directions possibles ;
- initialiser un dictionnaire qui associera à chaque direction possible la moyenne des scores de N parties jouées aléatoirement ;
- pour chaque coup possible, à partir de la position courante :
  - copier la grille ;
  - jouer les N parties avec une obtention des coups qui est aléatoire
  - calculer la moyenne de ces scores ;
  - remplir le dictionnaire ;

— afficher la moyenne des scores de chaque coup et choisir celui qui a la plus grande moyenne ;
— retourner ce choix.

Testez votre fonction en lançant le script.

Voici un exemple d'affichage que vous obtenez (en fin d'exécution du programme) :

```
Les scores de chaque direction :
 4 (Gauche) = 17919.73
 6 (Droite) = 17896.67
Je joue 4 (Gauche)
+------+------+------+------+
|      |      |      |      |
|  4   |  8   |  4   |      |
|      |      |      |      |
+------+------+------+------+
|      |      |      |      |
|  8   |  32  |  2   |      |
|      |      |      |      |
+------+------+------+------+
|      |      |      |      |
| 2048 |  16  |      |  4   |
|      |      |      |      |
+------+------+------+------+
|      |      |      |      |
|  16  |  8   |  4   |      |
|      |      |      |      |
+------+------+------+------+

Score : 19640
```