Collège LaSalle
Felix Soumpholphakdy

21 mars 2019
Structure de données avancées: 420-J13-AS

Projet Saison 4

Vue d'ensemble

Pour votre projet final, vous allez concevoir et mettre en œuvre des algorithmes d'intelligence artificielle (IA) permettant de jouer à des jeux simples. Ces algorithmes seront testés et notés par rapport à des algorithmes factices ainsi qu'aux soumissions d'autres étudiants. Votre intelligence artificielle sera simple au début, mais à chaque itération, elle doit s'améliorer et s'adapter aux nouvelles exigences et aux adversaires de plus en plus difficiles.

La saison finale vise à concevoir une forte intelligence artificielle capable de vaincre les adversaires créatives. Les élèves apprendront le modèle de chaîne de Markov comme stratégie de prédiction. Les étudiants qui souhaitent se défier davantage peuvent tenter de battre les autres élèves dans le classement pour obtenir des points bonus.

Distribution des notes

Jeu 1 (RPSLS)	10%
Jeu 2 (à annoncer plus tard)	10%
Rapport	10%
Total	**30%**

Saison 4 - Date limite de la soumission est le 29 mars (bonus)

Instructions

1. Cloner le projet à https://github.com/felixsoum/420J13AS-RPSLS
2. Recherchez le fichier qui vous est attribué. Il sera situé dans le chemin: `\RPSLS\AI\S{numéro de section}\{4 lettres du code permanent}.cs`
3. Fix any problems from the previous week.
4. Substituez (override) et implémentez la méthode Observe() qui permet à votre IA d'observer le coup précédent de votre adversaire.
5. Mettez à jour votre méthode Play() pour prendre en compte ces observations afin de surmonter les défis liés à l'IA.
6. Configurez votre IA en allant sur Program.cs, en remplaçant RandomAI par 15, à la ligne 15, puis en exécutant le programme (Ctrl + F5).

```
var game = Game.Create<RandomAI>();
```

7. Soumettez seulement votre dossier de classe à Lea (XXXX.cs).

Règles

La règle la plus importante est de ne pas plagier. Il est recommandé et bon de travailler avec des camarades de classe pour discuter de stratégies et de pratiques opposées, mais copier-coller le travail de quelqu'un d'autre est du plagiat. En tant que informaticiens, nous avons conçu de très bons outils pour détecter cela dans le code, même si les variables sont renommées ou les instructions réorganisées. Vous pourriez même apprendre quelques algorithmes pour cela dans ce cours. En gros, ne faites pas cela ou vous serez fortement pénalisé. De plus, vous ne pouvez pas dépendre d'autres classes d'intelligence artificielle (n'essayez pas d'invoquer les méthodes d'un autre). Vous serez également pénalisé si vous faites cela.

Les règles suivantes sont des directives pour le tournoi. Si vous ne les suivez pas, vous serez disqualifié, mais vous ne serez pas pénalisé pour les notes du projet.

- De nouvelles règles ne seront ajoutées qu'après la fin d'un tournoi et suite à la détection de comportements indésirables.
- Jouer un coup illégal entraînera la perte du tour.
- Si vous avez besoin d'un nouvel objet aléatoire, vous êtes uniquement autorisé à utiliser Game.SeededRandom, qui est initialisé car les résultats du jeu doivent être reproductibles (déterministes).
- Chaque tour de votre IA doit prendre moins de 10 millisecondes pour être exécuté. Un tour consiste en un appel à Observe() puis à Play(). Si votre IA prend trop de temps, le système vous le dira dans la console. Ce délai est sujet à changement.
- Vous n'êtes pas autorisé à utiliser les classes de l'espace de noms System.Reflection, car vous n'en avez pas besoin et si vous le faites, je me méfie de votre code.
- Si vous essayez d'instancier vous-même une intelligence artificielle, vous obtiendrez une exception non autorisée.

Défis

Chaque défi met votre IA contre une IA factice spécifique. Un défi est composé de 20 batailles et chaque bataille est composée de 100 rondes de RPSLS. Pour réussir le défi, votre IA doit vaincre le mannequin dans toutes les batailles, mais pas nécessairement à toutes les rondes. Une bataille est gagnée par l'IA qui a remporté le plus grand nombre de rounds. Entre chaque combat, la graine aléatoire (random seed) changera. La graine initiale (initial seed) sera également modifiée avant d'évaluer vos IA. Cela rend impossible à une IA de réussir un défi en raison de la chance ou du codage difficile de tous les mouvements. Vous pouvez basculer les défis simplement en décommentant ou en commentant le code (Program.cs, ligne 17-19). Vous pouvez également exécuter plusieurs défis de manière séquentielle.

```
game.Challenge<MarkovOneAI>();
game.Challenge<MarkovTwoAI>();
game.Challenge<ShakespeareAI>();
```

Classement

Un tournoi est mis en place pour encourager les étudiants à se surpasser. Les meilleurs élèves de chaque section auront la possibilité de présenter de manière informelle leur intelligence artificielle à la classe afin de recevoir des points bonus. N'hésitez pas à essayer le tournoi vous-même en exécutant l'instruction game.PlayTournament() dans la ligne 21 de Program.cs. Notez que si vous souhaitez obtenir le même résultat que le tournoi réel, veillez à désactiver (commenter) les défis, car ils affecter le résultat par l'utilisation de l'objet SeededRandom. La graine (seed) sera tirée des numéros de loto hebdomadaires

```
game.PlayTournament();
```

Le pointage

[1 point] Pour avoir réussi le défi contre *MarkovOneAI*. Pour chaque coup *x*, il existe un coup *y* qui a une forte probabilité (60%) d'être joué juste après *x*. Par exemple, après avoir joué un Rock, l'IA pourrait préférer jouer un Paper la plupart du temps.

[1 point]
Pour avoir réussi le défi contre *MarkovTwoAI*. Pour chaque coup *x* et *y*, il existe un coup *z* qui a une forte probabilité (60%) d'être joué juste après *x*, puis *y*. Par exemple, après avoir joué un Rock, puis un Paper, l'IA pourrait préférer jouer un Scissors la plupart du temps. Mais il est possible qu'après avoir joué un Spock, puis un Paper, l'IA préfère jouer un Lizard.

[1 point]
Pour avoir réussi le défi contre *ShakespeareAI*. Cette IA convertit le SonnetXVII de Shakespeare en une séquence de coups. Cette séquence de coups est disponible via la méthode statique ShakespeareAI.CreateSequence (). Ensuite, une fois instanciée, l'IA choisira un index aléatoire et la lecture se déplacera séquentiellement à partir de cet index. Notez que Shakespeare a une mauvaise mémoire, il y a donc 50% de chance que, au lieu de jouer le coup correct, il joue un coup aléatoire.