Apprentissage Automatique, Mini-projet

M1, année 2016-2017

Mini-projet : réseau de neurones appliqué au "jeu des bâtons"

1 Introduction

Le but de ce TP est d'apprendre à la machine comment jouer au mieux au "jeu des bâtons". Il existe en effet une "technique" pour gagner en fonction du nombre de bâtons à jouer et de quel joueur commence la partie. Nous allons utiliser une méthode de réseau de neurones simple (type perceptron) pour que la machine apprenne à jouer et découvre ainsi la "technique".

Dans notre programme, la machine pourra jouer selon trois modes :

- "easy" : elle joue aléatoirement 1, 2 ou 3 bâtons (dans la limite des règles, cf. ci-dessous)
- "medium" : elle joue aléatoirement sauf à la fin où elle ne commet pas d'erreurs évidentes
- "hard" : elle joue via son réseau de neurones grâce à l'apprentissage

Le travail va se diviser en trois phases :

- prise en main du code existant, premiers tests de jeu
- phase d'apprentissage : on va entrainer le réseau de neurones en jouant un grand nombre de fois
- phase de tests : on va créer plusieurs scénarii pour observer les comportements de la machine en fonction de plusieurs paramètres et notamment vérifier si la machine joue bien parfaitement en mode "hard".

2 Travail à rendre avant le vendredi 5 mai au soir

N'oubliez pas de préciser vos noms/prénoms et d'envoyer votre archive à : julien.grosjean@chu-rouen.fr. L'archive contiendra :

- les classes Python complétées et commentées
- des scripts Python correspondant aux différents scénarii
- un petit rapport avec les réponses aux questions posées ci-dessous

3 Règles du "jeu des bâtons"

Sur un plateau sont disposés quinze bâtons (le nombre peut varier mais nous prendrons cette valeur durant tout le projet). Deux joueurs s'affrontent et doivent, chacun leur tour, prendre un, deux ou trois bâtons. Celui qui prend le dernier bâton a perdu.

4 Présentation des classes Python

Vous disposez de six classes réparties dans trois fichiers :

- la classe `Game` contient l'algorithme du déroulement d'une partie du jeu des bâtons. Son constructeur prend en paramètre le nombre de bâtons au début du jeu. La méthode `start` permet d'effectuer une partie. NB : il ne sera normalement pas nécessaire de modifier cette classe.
- la classe `Player` est la super-classe représentant un joueur. NB : il ne sera normalement pas nécessaire de modifier cette classe.
- la classe `HumanPlayer` est utilisée pour faire jouer un joueur humain au jeu des bâtons. NB : il ne sera normalement pas nécessaire de modifier cette classe.
- la classe `CPUPlayer` est utilisée pour faire jouer la machine au jeu des bâtons. Il faudra compléter le code de cette classe.
- la classe `NeuronNetwork` représente un réseau de neurones, c'est-à-dire un ensemble d'instances de classe `Neuron`. NB : il ne sera normalement pas nécessaire de modifier cette classe.
- la classe `Neuron` représente le neurone de base d'un réseau de neurones. Chaque neurone contient des connexions vers d'autres neurones. Il faudra compléter le code de cette classe pour que le réseau de neurones fonctionne.

5 Les bases et le mode simple

Q. Se familiariser avec le code python en l'analysant rapidement et en commentant les classes.
Q. SCRIPT 1 : Créez un script python en important les classes nécessaires et permettant d'effectuer une partie entre un joueur humain (vous !) et la machine (mode de jeu "easy", nombre de bâtons = 15 et mode "verbose" à `True`).
Q. En utilisant ce script, commencez à jouer contre l'ordinateur en faisant quelques parties. L'ordinateur peut faire des erreurs évidentes lors du dernier tour ; à l'aide du code, expliquez pourquoi.

6 Mode intermédiaire

Q. Mode "medium" : écrire quelques lignes de code dans la méthode de l'IA (`playMedium`) pour éviter que l'ordinateur ne commette une erreur au dernier tour lorsqu'il a toutes les chances de gagner (le nombre de bâtons restants est de 2, 3 ou 4). Est-ce de l'apprentissage ? Expliquez.

7 Implémentation du réseau de neurones

Nous allons maintenant compléter le code d'utilisation du réseau de neurones. Pour cela, compléter les méthodes suivantes :

1. méthode `testNeuron` de la classe `Neuron`.
2. méthode `chooseConnectedNeuron` de la classe `Neuron`.
3. méthode `playHard` de la classe `CPUPlayer`.
4. méthode `recompenseConnection` de la classe `Neuron`.

Q. Testez bien l'utilisation du réseau de neurones en faisant jouer la machine d'abord contre vous (mode "hard", nombre de bâtons = 15 et mode "verbose" à `True`).

8 Apprentissage

Q. SCRIPT 2 : Pour être plus efficace dans l'apprentissage, nous allons maintenant laisser l'ordinateur jouer contre lui-même en mode "hard". Comment s'appelle cette méthode et pourquoi l'utilise-t-on ? Il faudra jouer plusieurs centaines de fois voire plusieurs milliers, essayez d'abord avec 800 parties en boucles en désactivant le mode "verbose" des parties.
Q. En laissant l'ordinateur jouer contre lui-même, que constatez-vous à la fin de N parties ? Pour cela, affichez les scores des deux joueurs et leurs réseaux de neurones respectifs à l'aide de la méthode `printAllConnections()` de `NeuronNetwork`.
Q. SCRIPT 3 : Affichez le nombre de parties gagnées par chaque joueur à la fin de toutes les parties. Expliquez la différence de scores entre les 2 joueurs. Refaire ces observations pour les autres modes et discutez ces résultats (faire les différents tests en faisant jouer la machine contre elle-même en mode "easy" contre "easy", "easy" contre "medium","medium" contre "easy", "medium" contre "hard", "hard" contre "medium", "easy" contre "hard", "hard" contre "easy" et "medium" contre "medium").

Q. Phase d'apprentissage : pour finir, jouez un très grand nombre de fois puis enregistrez le réseau neuronal dans un fichier sérialisé (éventuellement, aggrégez les deux réseaux neuronaux produits pour avoir un réseau neuronal final plus étoffé). Il est maintenant possible de jouer au jeu contre l'ordinateur sans la phase d'apprentissage.
Pour la sérialisation, utilisez le module `pickle` comme suit :

- sérialisation dans un fichier :
  `with open(fileName,'wb') as output: pickle.dump(neurons,output,pickle.HIGHEST_PROTOCOL)`
- désérialisation à partir d'un fichier :
  `with open(fileName, 'rb') as inp: ns = pickle.load(inp)`

9 Jeu final

Q. SCRIPT 4 : Créez un script correspondant au jeu "final" dans lequel on pourra demander le nom du joueur humain et le mode de jeu auquel il souhaite jouer. Charger le réseau de neurones au démarrage pour le mode "hard".
Q. Jouez en mode "hard" en se plaçant "Joueur 2" (joueur qui ne commence pas). Est-ce possible

de gagner ? Expliquer pourquoi.

10 Question optionnelle : évaluation

Q. Repartez du script de l'ordinateur contre lui-même (phase d'apprentissage) et évaluez le taux d'erreur en jouant un grand nombre fois. Décrivez le protocole pour calculer ce taux et mettez-le en pratique. Vous pourrez essayer de visualiser à quel moment la machine ne fera quasiment plus aucune erreur.
En vous aidant de ceci et des autres paramètres/méthodes utilisés dans ce réseau de neurones, proposez une (ou plusieurs) solution(s) pour améliorer le réseau neuronal final.