TP : Réseau de Neurones

Objectif du TP

L'objectif de cette séance est d'utiliser un réseau de neurones pour contrôler Ms. Pacman en java en utilisant la librairie Encog (https://www.heatonresearch.com/encog/).

EXERCICES SUR MACHINES !

Comme le disent Yannakakis et Togelius, les auteurs du livre « AI in Games », dans le paragraphe 2.5.1.7, le plus simple pour mettre en œuvre un réseau de neurones pour Ms. Pacman est d'apprendre comment des experts ont joué à Ms. Pacman. Le problème est que nous ne disposons pas de parties jouées par des experts… pas encore ! Le plan prévu pour ce TP est le suivant :

- Faire jouer un pacman expert et enregistrer dans un fichier de données l'état du jeu et les décisions qu'il prend ;
- Construire et entrainer en off-line un réseau de neurones avec les données précédemment enregistrées ;
- Utiliser le précédent réseau de neurones en on-line pour décider quelle direction prendre.

Mais avant cela, je vous propose de découvrir une nouvelle librairie permettant de manipuler des réseaux de neurones.

Exercice 1 Découverte de la librairie Encog

Bien évidemment, Encog est loin d'être la seule librairie de Machine Learning. Nous devrions plutôt utiliser une autre librairie comme DeepLearning4J mais l'intérêt de la librairie proposée est sa simplicité. Il propose également les algorithmes Neat et HyperNeat utilisés dans le module M422 qui consiste à faire évoluer des réseaux de neurones par algorithme génétique (Neuro-évolution).

1. Téléchargez les différents jar de la librairie à cette adresse https://github.com/encog/encog-java-core/releases et ajoutez les à votre projet Eclipse.
2. Construisons un réseau de neurones qui apprend la table de vérité de XOR avec deux variables binaires. Pour cela, téléchargez le code `testEncog.java` et exécuter le plusieurs fois. Que remarquez-vous ? Ajoutez un paramètre à la méthode `reset` et recommencez.
3. Modifier le programme précédent pour que le réseau apprenne la table de vérité XOR de 3 variables.
4. Imaginons, juste un instant que 1 XOR 1 XOR 1 = 3. Modifiez ce que vous avez fait pour que le réseau apprenne cette fausse table de vérité. Lancez l'apprentissage. Que constatez-vous ? Qu'en déduisez-vous ?
5. Modifiez la boucle d'apprentissage de telle sorte que l'apprentissage s'arrête après MAX_ITERATION sans amélioration de l'erreur.
6. Revenez à la vraie table de vérité et augmentez le nombre de neurones dans la couche cachée. Qu'observez-vous ?
7. Ajoutez une nouvelle couche cachée et comparez avec le précédent réseau.

Exercice 2 Création du jeu de données

1. Selon vous, quelles pourraient être les entrées et les sorties du réseau de neurones ? Indice : essayez d'être le plus généraliste possible, i.e. être le plus indépendant possible de la position de pacman.

2. Créez un nouveau contrôleur de pacman appelé `ANNPacman` (classe qui hérite de `Controller<MOVE>`).
3. Dans cette classe, créez une nouvelle méthode de classe (`static`) nommée `ANNInputState` qui va calculer les entrées du réseau de neurones et qui renvoie un dictionnaire <Nom de la donnée, Valeur réelle de la donnée> à partir de la partie en cours (`Game`). Indice : pensez à normaliser vos données entre 0 et 1.
4. Dans la classe `Executor`, inspirez-vous de la méthode `runGameTimedRecorded` pour créer une nouvelle méthode `createData4ANN` qui créera le fichier texte (au format csv) des données utilisées ultérieurement par le réseau de neurones. Pour éviter d'avoir trop de données, il est conseillé de sauver l'état du jeu uniquement quand Pacman change de direction ou qu'il arrive sur un nœud de jonction.
5. Créez le fichier de données en appelant la méthode précédente avec le bot le plus intelligent que vous ayez (`StarterPacman` si rien de mieux) avec les fantômes les plus intelligents (`AggressiveGhosts`).

Exercice 3 Création de ANNPacman

1. Définissez le constructeur de la classe `ANNPacman`. L'objectif est de créer un réseau de neurones et de le faire apprendre sur les données créées dans l'exercice précédent.
2. Définissez le comportement du pacman dans la méthode `getMove` en utilisant le réseau de neurones entrainés dans le constructeur.
3. Maintenant que vous avez programmé tout le processus, vous pouvez améliorer votre bot en
    a. Augmentant le jeu de données d'apprentissage en enregistrant plusieurs parties ;
    b. Modifiez / améliorez les données apprises pour chaque état du jeu
    c. Modifiez la structure du réseau de neurones en réseau de neurones récurrents (qui sait mieux apprendre les données temporelles, ce qui est le cas ici). Ce type de réseaux est prévu dans Encog avec la classe `ElmanPattern` (http://www.javased.com/?api=org.encog.neural.pattern.ElmanPattern).

Sources pour ce TP :

.