Projet apprentissage par renforcement - Pacman 2ème partie

Olivier Goudet

March 17, 2022

Tout d'abord, télécharger puis décompresser le dossier *PacManEtud2_ 2022* disponible sur Moodle dans l'onglet "Source projet Pacman - DeepQlearning - 2ème partie".

Importer le projet *PacManEtud2_ 2022* avec Eclipse en cliquant sur File -> import -> Existing Maven Projects.

Gérer le *build path* du projet en ajoutant la librairie *commons-lang3-3.12.0.jar* qui est contenue dans le dossier en tant qu'*external jar* pour votre projet.

Implémentation des stratégie d'apprentissage avec réseau de neurones

Pour tester ces méthodes d'apprentissage il faudra utiliser le script de lancement *main_ batchMode*.

1. Compléter la classe *ApproximateQLearningStrategy_ withNN* qui correspond à l'algorithme de ApproximateQLearning avec réseau de neurones présenté dans la première partie du cours 4. Inspirez vous du TD 4. Pour l'encodage des features vous pouvez reprendre exactement la même fonction que celle utilisé dans la première partie du projet dans la classe "ApproximateQLearningStrategy.java.
2. Comment varient les résultats en fonction des features choisies ?
3. Compléter la classe *DeepQLearningStrategy* qui correspond à l'algorithme de DeepQlearning avec réseau de neurones présenté dans la deuxième partie du cours 4. Inspirez vous du TD 4.
4. Faites varier la range vue par le pacman autour de lui et évenutellement le nombre de couches dans le réseau de neurone. Comment évoluent les résultats ?

Fonctions du jeu pacman utiles pour la réalisation des algorithmes

- Dans la classe **PacmanGame** vous avez accès à l'agent pacman qui est caractérisé par une position de type **PositionAgent** avec ses coordonnées en X et en Y.
- Dans la classe **PacmanGame** vous avez accès à la liste des agents fantômes avec leurs positions (fonction getPostionFantom()).
- Dans la classe **PacmanGame** vous avez accès à l'attribut labyrinthe de type **Maze** qui dispose des méthodes *isCapsule()* et *isFood()* qui permettent de savoir si une capsule ou bien une pacgomme se trouve en position (x,y).
- Dans la classe **PacmanGame** vous avez accès à une méthode *isLegalMove* qui permet de savoir si un agent peut réaliser une action donnée. Pour améliorer la performance de l'algorithme, il est en effet conseillé de considérer uniquement des actions légales pour le pacman dans vos algorithmes.
- Dans vos stratégies d'apprentissage, il est préférable d'adapter les choix des actions au mode *train* ou *test*. Notamment, lorsqu'on est en mode *test*, on réglera la valeur d'epsilon à 0 (sans choix aléatoire). Ceci permettra de mieux évaluer et comparer vos différentes stratégies.