Rapport de projet

SMA – Apprentissage par renforcement

Alban FLANDIN p1504017

Lucas MAGNANA p1503046

Systèmes Multi-Agents

Année universitaire 2019-2020

Introduction

Dans le cadre de notre UE Multi-agents and Self-Systems, nous avons été amenés à implémenter des techniques d'apprentissage par renforcement sur l'exemple du jeu Pacman. Ce rapport fait état de nos choix de modélisation.

TP1 – Partie 5.1

En passant le bruit à 0, il trouve une politique optimale en se dirigeant en permanence à droite. C'est logique, si la direction qu'il choisit ne peut pas être différente de celle qu'il prendra effectivement, il est sûr de gagner en suivant le chemin jusqu'à l'état absorbant gagnant.

TP1 – Partie 5.2

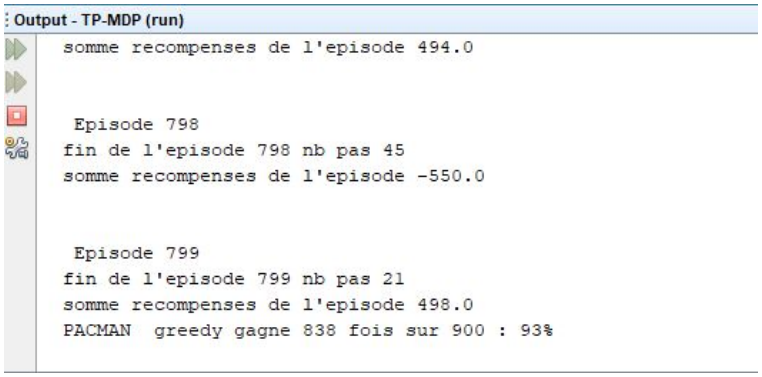
Pour obtenir une politique optimale vers l'état à 1.0 en prenant le chemin risqué, on réduit le bruit à 0, et on passe les autres récompenses à -5 (pour que continuer vers l'état à 10 ne vaille pas le coup).
Pour obtenir un chemin risqué vers l'état absorbant de 10.0, on réduit le bruit à 0 et on passe les autres récompenses à -1 (pour que prendre un chemin long ne soit pas optimale).
Pour éviter les états absorbant, il faut augmenter les autres récompenses à une valeur supérieure à 10.

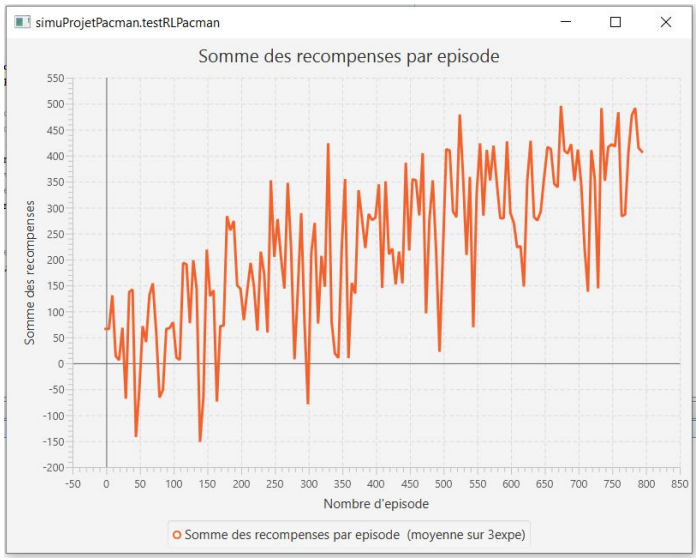
Partie 2.2

Cette partie porte sur la modélisation des états du jeu dans une approche de Q-Learning tabulaire. Le but était d'isoler les paramètres qui, s'ils sont similaires pour 2 états, font qu'on peut considérer ces deux états similaires eux-aussi. Après plusieurs essais de différents paramètres (modéliser l'environnement proche du pacman, la direction dans laquelle se trouve la pièce la plus proche, etc.), la combinaison de paramètres la plus efficace au moins sur les petites grilles a été la suivante :

- Les positions X et Y du Pacman
- Un tableau contenant les positions du ou des fantomes
- Un tableau contenant les positions des différentes « food »

Cette approche est particulièrement efficace pour les petites grilles, qui comportent peu de pièce. En 500 épisodes d'apprentissage, on obtient des taux de réussite supérieurs à 90%.





Toutefois, cette approche atteint vite ses limites. En effet, sur la grille medium, le très grands nombre de food présent entraine une multiplication des états possibles, et demanderait beaucoup plus d'apprentissage pour devenir viable. Ici, sur 500 épisodes d'apprentissage, on atteint rarement plus de 5% de victoires.

Partie 2.3

Ici, le but est d'améliorer la généralisation des états en approximant la fonction Q. La première approche avec la matrice d'identité nous donne des résultats à peu près équivalent à la première partie sur les small grid. A cause du temps d'exécution beaucoup trop grand (dû à un trop grand nombre d'états possibles), nous n'avons pas pu le tester sur la medium grid.

Pour la classe FeatureFunctionPacman avec les 4 paramètres demandés, les résultats sont bien meilleurs sur la medium grid (plus de 90% de victoires). Nous avons aussi remarqué lors de nos tests que la convergence est beaucoup plus rapide : même lorsque nous étions à 50% de victoires (à cause de quelques erreurs dans notre code), augmenter le nombre d'épisodes d'apprentissage ne changeait plus du tout le taux de victoire. Il atteignait très rapidement son « taux maximal d'apprentissage ». Cela est confirmé par la courbe, qui montre un apprentissage très rapide.