RAPPORT APPRENTISSAGE ARTIFICIELLE
2021 – 2022

Nom : TOURE
Prénom : Boubacar
Numéro étudiant : 18 00 94 22

Niveau : Master 1 informatique
Date : 09 Avril 2022 à Angers
Professeur : Monsieur Olivier GOUDET
Cours : Apprentissage Artificielle
Objectif : Implémenter les premières stratégies d'apprentissage pour le *Pacman* afin de lui permettre d'agir de manière précis et réfléchie grâce à l'intégration d'un système d'apprentissage automatique mise en place par des algorithmes d'Intelligence Artificielle.

FACULTÉ
DES SCIENCES
UNIVERSITÉ D'ANGERS

SOMMAIRE

I. **Introduction**
II. **Stratégies d'apprentissages**

I.Introduction

Dans le cadre de ce projet, nous avons pour but d'implémenter des algorithmes d'apprentissages artificielles afin de permettre aux agents ***Pacman*** de pouvoir prendre des décisions et d'apprendre des erreurs qu'ils commentent aux fur et à mesures que le jeu avance.

Pour la réalisation de cette expérience, nous avions en notre disposition un projet déjà préalablement implémenté. On avait pour responsabilité de faire évoluer la partie concernant l'IA et ainsi rendre le jeu *Pacman* plus attractif, continue et évolutif.

Pour chaque méthode d'apprentissage implémentée, il nous a été fourni des fichiers sources permettant de lancer l'application via l'interface graphique en mode :

- Debug (main_debugMode)
- Standard (main_standardMode)

Pour pouvoir tester ces différentes méthodes, nous utiliserons le script de lancement main_batchMode.

II. Stratégies d’apprentissages

Au cours de la mise en pratique de notre projet, nous avons appliqué, en tout deux (2) méthodes d’apprentissages. Des séries de tests ont été réalisé sur ces différents algorithmes dont nous vous exposerons les détails. Parmi ces différentes algorithmes que nous avons implémenté, nous avons :

- **TabuLarQLearning**

A l’aide de l’algorithme **TabuLarQLearning,** nous avons remarquer que l’agent *Pacman* apprend au fur et à mesure que le jeu avance. Il retient chaque bonne et mauvaise action qu’il a eu à effectuer, ce qui lui donne la possibilité d’éviter certaines erreurs qu’il a commis précédemment.

Cette méthode possède des inconvénients car plus le champ d’action sur le terrain de jeu est grand, plus cet algorithme requiert de la mémoire ainsi que de l’espace sur notre système.

Pour pallier à ce manquement, j’ai eu à implémenter un Hashtable contenant des variables de types string afin de représenter les différents états de notre terrain de jeu. Ce qui nous permettrait d’effectuer au minimum un certain de nombre de tour de jeu même sur un terrain de grandes tailles car notre algorithme nous permettra de stocker en mémoire que l’espace dont il a besoin grâce à la capacité des hashtables qui leur permettent d’allouer dynamique de la mémoire.

- **ApproximateQLearningStrategy**

A l’aide de l’algorithme **ApproximateQLearningStrategy,** l’approche est totalement différente. Pour la mise en place de cet algorithme, on procède à une extraction des caractéristiques pour caractériser un couple. Contrairement au **TabuLarQLearning**, on utilise un vecteur de poids permettant de prédire les meilleures actions à effectuer. Cette approche nous permet de gérer au mieux la quantité d’espace pouvant être alloué par notre programme. Ce qui rend cet algorithme plus apte à faire des tests sur des terrains de jeu de grandes envergures. L’objectif de cette approche est d’ajuster les paramètres W du modèle Qw de façon à minimiser l’erreur(W). Il permet aussi de gérer des états différents et non encore explorés peuvent partager des caractéristiques communes « fi » avec des états déjà explorés. De plus, Les connaissances qu’on a appris sur des états déjà explorés peuvent être utilisées pour traiter des états non explorés. Le comportement est plus robuste : on fait des choix similaires pour des états similaires.

Le seul inconvénient dans cette méthode est qu’il demande une extraction des features fi (s, a) à chaque itération et pour chaque action possible, ce qui peut être coûteuse. De plus, la qualité du programme dépend des différentes features qui seront implémentées. Plus les features seront performants, plus votre algorithme sera puissant et précis.