TP2 : Algorithmes itératifs pour les processus de décision Markovien

DJAGHLOUL AYOUB

30/03/2024

1 Introduction

Dans ce TP2 sur les algorithmes itératifs pour les processus de décision Markovien (MDP), l'objectif est d'appliquer l'algorithme d'itération de la valeur pour déterminer la politique optimale dans un environnement de grille (gridworld). Ce problème met en évidence l'application des principes des processus de décision Markoviens à un environnement simple mais illustratif.

2 Définitions des fonctions

Dans cette section, nous définissons les fonctions nécessaires à notre algorithme d'itération de la valeur pour résoudre le problème du processus de décision markovien dans un environnement de grille. Ces fonctions comprennent :

- **move(state, action)** : Cette fonction retourne les coordonnées de l'état après avoir effectué une action.
- **is_valid_move(position, action)** : Cette fonction vérifie si un mouvement proposé est valide, c'est-à-dire s'il ne sort pas des limites de la grille et ne traverse pas un mur.
- **possible_actions(position)** : Cette fonction retourne une liste des actions valides à partir d'une position donnée, en utilisant la fonction *is_valid_move* pour filtrer les actions non valides.
- **action_90degree(action, actions_possible)** : Cette fonction identifie les actions perpendiculaires à une action donnée.
- **get_state_index(x, y)** : Cette fonction retourne l'indice linéaire d'un état dans la grille.

3 Initialisation

Dans cette section, nous initialisons les paramètres et les variables nécessaires à notre algorithme :

- **n_rows, n_cols** : Les dimensions de la grille.
- **actions** : Les actions possibles dans l'environnement (mouvement vers le haut, le bas, la gauche et la droite).
- **goal** : Les coordonnées de l'état objectif.
- **bad** : Les coordonnées de l'état pénalisant.
- **wall** : Les coordonnées du mur dans la grille.
- **n_states** : Le nombre total d'états dans la grille.
- **gamma** : Le facteur d'escompte pour les récompenses futures.
- **threshold** : Le seuil de convergence pour l'algorithme.
- **V** : La table des valeurs des états.
- **reward** : La table des récompenses pour chaque état de la grille.

4 Itération de la Valeur

Dans cette section, nous itérons sur chaque état de la grille pour mettre à jour la fonction de valeur jusqu'à convergence. L'algorithme s'exécute comme suit :

1. Pour chaque état de la grille, à l'exception des murs, de l'état objectif et de l'état pénalisant, nous calculons les récompenses attendues pour chaque action possible en tenant compte de l'action principale et des déviations possibles.
2. Nous calculons la somme des récompenses pour chaque action, en ajustant les probabilités en fonction du nombre d'actions de déviation possibles.
3. Nous mettons à jour la fonction de valeur pour chaque état en prenant le maximum des récompenses attendues.
4. L'algorithme continue d'itérer jusqu'à ce que le changement maximal dans les valeurs des états entre deux itérations consécutives tombe en dessous du seuil de convergence.

5 Politique Optimale π^*

Dans cette section, nous déterminons la politique optimale π^* qui indique la meilleure action à prendre à chaque position de la grille. Pour cela, nous utilisons la fonction de valeur optimale calculée précédemment :

1. Pour chaque état de la grille, à l'exception de l'état objectif, de l'état pénalisant et des murs, nous déterminons la meilleure action à prendre en calculant la valeur attendue de chaque action possible.
2. Nous mettons à jour la politique optimale en associant à chaque état la meilleure action.
3. Enfin, nous remplaçons les coordonnées de l'état objectif, de l'état pénalisant et des murs par des étiquettes dans la politique optimale pour une meilleure lisibilité.

6 Résultats

Après avoir exécuté l'algorithme, nous obtenons la fonction de valeur optimale ainsi que la politique optimale π^*, qui nous indique la meilleure action à prendre à chaque position de la grille.

6.1 Fonction de Valeur Optimale

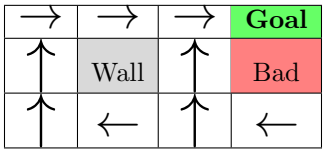
La fonction de valeur optimale obtenue est la suivante :

$$\begin{bmatrix} 0.60 & 0.69 & 0.79 & 1.0 \\ 0.52 & 0 & 0.47 & -1.0 \\ 0.45 & 0.39 & 0.34 & 0.05 \end{bmatrix}$$

Cette matrice représente les valeurs estimées pour chaque état de la grille.

6.2 Politique Optimale π^*

La politique optimale obtenue est la suivante :



Cette matrice indique la meilleure action à prendre à chaque position de la grille.

7 Conclusion

Enfin, cette expérience nous a également confrontés à des défis pratiques tels que le choix des paramètres d'algorithmes, la gestion des états terminaux et la visualisation des résultats. Ces défis nous ont permis de développer nos compétences en résolution de problèmes et en programmation.

Dans l'ensemble, cette première expérience en reinforcement learning a posé les bases pour des explorations futures dans ce domaine fascinant. En continuant à étudier et à expérimenter avec différentes techniques et applications de RL, nous pourrons approfondir notre compréhension et développer des solutions innovantes pour des problèmes complexes de prise de décision.