ULB UNIVERSITÉ LIBRE DE BRUXELLES

INFO-F311 - Projet d'IA 3

Apprentissage par renforcement

Auteur:
Manuel Rocca - 000596086

Professeurs:
Tom Lenaerts
Assistants:
Axel Abels
Martin Colot
Yannick Molinghen
Pascal Tribel

Année académique 2025-2026

Contents

1 Introduction

Pour ce troisième projet du cours d'Intelligence Artificielle, nous avons implémenté des algorithmes d'apprentissage par renforcement (*reinforcement learning* en anglais). Nous utilisons toujours l'environnement *LLE* comme pour les parties 1 et 2.

Trois différences (ou ajouts) sont introduites: les cases tourbillons (où l'agent meurt) la fonction de récompense, et surtout, l'aspect stochastique, non-déterministe de l'environnement.

1.1 Environnement stochastique

Un environnement stochastique est un environnement où, lorsque l'agent effectue une action, nous ne sommes pas certains de l'état dans lequel il va arriver. En pratique, si un agent dans un état s souhaite effectuer une action a pour arriver dans un nouvel état s', il y a une probabilité p qu'une autre action aléatoire soit prise.

1.2 La fonction de récompense

Une fonction de récompense est une fonction qui, pour chaque transition d'état, octroie une valeur. En d'autres termes, une valeur numérique est associée à chaque action prise par l'agent afin de le diriger de manière idéalement souhaitée.

Associer une valeur adéquate est tout sauf quelque chose de simple. Des comportements inattendu peuvent survenir si cette fonction est mal définie (par exemple: si un agent doit atteindre une sortie avec une récompense positive mais que le coût de vie, le coût par pas est très élevé, il aura plus tendance à chercher un moyen de faire le moins de pas possible en trouvant un endroit plus proche pour mourir permettant une minimisation de son score total).

2 Value Iteration

L'algorithme *Value Iteration* utilise l'équation de Bellman (1) de manière itérative pour mettre à jour les valeurs estimées de chaque état de l'environnement.

$$V^*(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right] \tag{1}$$

Figure 3: Équation de Bellman caractérisant les valeurs optimales

Les itérations de l'algorithme sont bornées par une valeur δ comme suit:

$$\max_{s \in S} |V_{k+1}(s) - V_k(s)| < \delta \tag{2}$$

En d'autres termes, à chaque itération, la différence/variation entre les valeurs des états précédents et les nouvelles valeurs est calculée en appliquant l'équation 1. Si celle-ci est inférieure du seuil δ donné, l'algorithme s'arrête.
L'algorithme converge après un certain nombre d'itérations. Ce nombre peut être, dans certains cas, très grand voire quasi infini. C'est pourquoi il est intéressant d'établir un seuil de variation maximal afin d'arrêter l'algorithme après une durée d'exécution raisonnable. Comme nous l'avons vu au cours théorique, parfois un dixième des itérations suffisent à obtenir des valeurs pratiquement égales aux valeurs obtenues à la convergence.

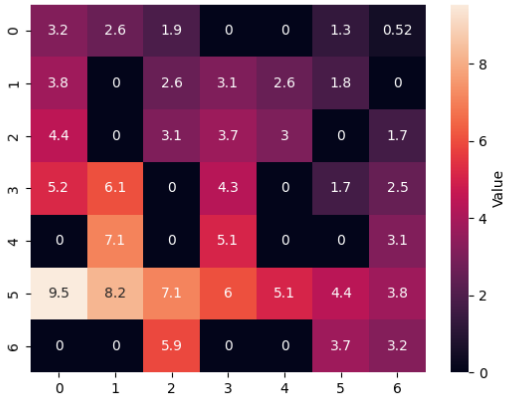
2.1 Entraînement de l'algorithme

Dans cette section, nous nous intéressons aux valeurs d'états finales obtenues par notre implémentation de l'algorithme *Value Iteration* pour des valeurs de $\delta \in \{1, 0.1, 0.01, 0.005, 0.001\}$ ainsi que le nombre d'itérations k. Pour chaque valeur de δ testée, nous usons comme facteur de réduction $\gamma = 0.9$ et la probabilité de l'environnement à prendre une autre action que celle souhaitée $p = 0.1$[1].

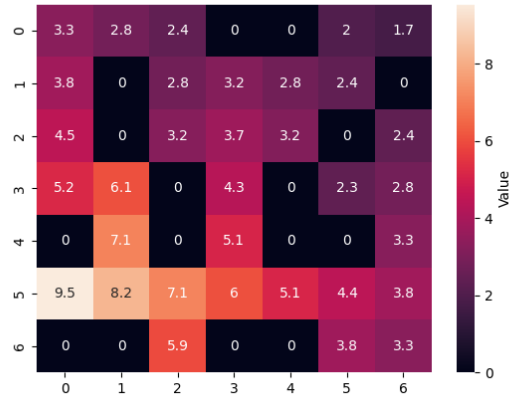
δ	k
1	10
0.1	14
0.01	18
0.005	18
0.001	20

Table 1: Nombre d'itérations de l'algorithme *Value Iteration* en fonction d'un δ donné.

Plus il y a d'itérations, plus nous nous approchons de la convergence. Voici les heatmaps pour $\delta = 1$ et $\delta = 0.001$ affichant les valeurs d'états dans l'environnement stochastique après k itérations:



(a) $\delta = 1$, $k = 10$



(b) $\delta = 0.001$, $k = 20$

Nous nous rendons très vite compte que la valeurs des états les plus éloignés de l'état initial en $(0, 6)$ (en haut à droite) convergent plus rapidement que ceux plus proches de l'état d'origine. Cela s'explique par le fait que toutes les valeurs d'états calculées par *Value Iteration* auront une valeur nulle tant qu'une récompense n'a pas été trouvée. Démontrons cela de manière concise:

Démonstration. En observant l'équation 1, il est clair que la valeur est nulle dans deux cas:

- si $T(s, a, s')$ est nulle. Or cette valeur n'est jamais nulle, il y a toujours une probabilité donnée de passer d'un état s à un état s' suivant.
- si $R(s, a, s') + \gamma V^*(s')$ est nul. Initialement tous les $V(s)$ sont nuls, donc le seul terme qui peut donner une valeur non-nulle à cette expression est la récompense $R(s, a, s')$ (γ est une constante non-nulle).

En somme, les valeurs d'état calculées se propagent à partir des transitions accordant une récompense expliquant ainsi la convergence plus rapide dans ces environs. □

[1] Les valeurs suivantes sont récoltées avec la seed `random.seed(0)`