FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

UNIVERSITÉ LIBRE DE BRUXELLES ULB

INFO-F-311: Intelligence Artificielle

Projet 3: Apprentissage par renforcement

Axel Abel **Martin Colot** **Tom Lenaerts**

Yannick Molinghen **Pascal Tribel**

1. Préambule

Dans ce projet, vous allez implémenter des algorithmes d'apprentissage par renforcement (reinforcement learning). On vous fournit des fichiers de base pour le projet que vous pouvez télécharger sur l'université virtuelle. Vous pouvez modifier tous ces fichiers à votre guise.

L'utilisation d'outils tels que ChatGPT est autorisée, et nous vous demandons d'expliquer brièvement comment vous les avez utilisés dans votre rapport.

2. L'environnement

Le problème que vous allez résoudre avec de l'apprentissage par renforcement est la carte de LLE illustrée dans la Figure 1. Dans cet environnement, les cases grises représentent des emplacements libres sur lesquelles l'agent peut se déplacer, les cases noires représentent des murs, et les tourbillons représentent des cases sur lesquelles l'agent peut marcher mais qui le font mourir en terminant l'épisode.

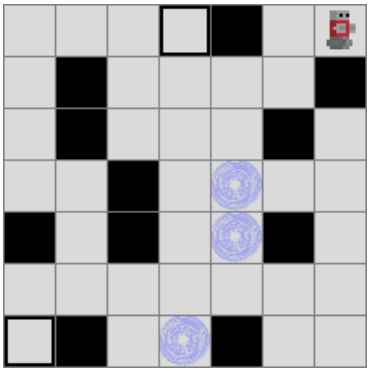

Figure 1. – Illustration du labyrinthe à résoudre

La case de départ de l'agent est la case située dans le coin supérieur droit de la carte. Le but de l'agent est de sortir du labyrinthe; soit par la sortie du haut (+1), soit par la sortie en bas à gauche (+10). L'agent peut mourir en marchant sur un tourbillon, auquel cas il meurt et reçoit une punition de −1. La fonction de récompense est représentée dans l'Équation 1.

$$R(s,a,s') = \begin{cases} 1 \text{ si } s' \text{ est la sortie supérieure} \\ 10 \text{ si } s' \text{ est la sortie inférieure gauche} \\ -1 \text{ si l'agent meurt dans un tourbillon} \\ 0 \text{ sinon} \end{cases} \tag{1}$$

2.1. Non-déterminisme

L'environnement que vous allez utiliser est non déterministe. En effet, lorsque l'agent effectue une action, il y a une probabilité p qu'une autre action aléatoire soit prise. La classe `Labyrinth` dans le fichier `env.py` vous permet d'instancier et d'interagir avec l'environnement. La fonction `random_moves` dans le fichier `main.py` vous donne un exemple minimal d'interaction.

3. Algorithmes

Vous devez implémenter l'algorithme d'itération de valeur value iteration et l'algorithme du Q-learning.

3.1. Value Iteration

L'algorithme de «Value Iteration» fonctionne de manière itérative, en mettant à jour à chaque itération son estimation de la valeur de chaque état grâce à l'équation de Bellman. On note V_k la fonction de valeur lors de la $k^{\text{ième}}$ itération de cet algorithme.

À chaque étape, l'algorithme calcule la valeur $V_k(s)$ de chaque état s à l'aide de l'Équation 2 dans laquelle $P(s, a, s')$ désigne la probabilité d'arriver en s' en prenant l'action a dans l'état s.

$$V_{k+1}(s) = \max_{a \in A} \sum_{s'} P(s, a, s')[R(s, a, s') + \gamma V_k(s')] \tag{2}$$

L'algorithme de Value Iteration prend en paramètre une valeur $\delta > 0$ qui détermine quand il faut s'arrêter. L'algorithme se termine dès que la variation d'une itération à l'autre est inférieure à δ, comme illustré dans l'Équation 3.

$$\max_{s \in S} |V_{k+1}(s) - V_k(s)| < \delta \tag{3}$$

Implémentez l'algorithme de Value Iteration dans le fichier `value_iteration.py`.

Notes:

- L'affichage avec `env.render()` est relativement lent par rapport au reste du code. Évitez d'y faire appel si vous n'en avez pas besoin.
- Quand vous implémentez l'algorithme, faites attention à vous baser sur V_k pour calculer V_{k+1} et à ne pas modifier V_k durant l'itération.
- N'oubliez pas que la valeur d'un état terminal est 0 par définition.
- Vous retrouverez dans main.py une fonction d'aide `plot_values` qui permet d'afficher une *heat map* qui représente la valeur de chaque état.

3.2. Q-learning

Le but de l'algorithme de Q-learning est d'apprendre une fonction $Q : S \times A \to \mathbb{R}$ qui associe une valeur réelle à chaque action dans chaque état. L'utilité de cette fonction Q est d'en dériver une stratégie optimale qui consiste à prendre l'action qui a la plus haute valeur dans chaque état.

Contrairement à Value Iteration, l'algorithme de Q-learning nécessite que l'agent interagisse avec l'environnement pour l'explorer et en découvrir le fonctionnement et mettre à jour la fonction Q comme illustré dans l'Équation 4 dans laquelle α est le facteur d'actualisation (*learning rate*) et γ est le *discount factor*. Notez que l'on peut calculer $V(s) = \max_{a \in A} Q(s, a)$.

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[R(s, a, s') + \gamma V(s')] \tag{4}$$

Vous devrez implémenter deux stratégies d'exploration: ε-greedy et Max-Boltzmann dans le fichier `qlearning.py`.

Exploration ε-greedy

Cette stratégie d'exploration consiste à choisir une action aléatoire avec une probabilité ε et à prendre l'action qui maximise $Q(s, a)$ avec une probabilité $1 - \varepsilon$, comme illustré dans l'Équation 5, où r est un nombre uniformément aléatoire entre 0 et 1.

$$\pi(s) = \begin{cases} \arg\max_{a \in A} Q(s, a) & \text{si } \ r \leq \varepsilon \\ a \sim A & \text{sinon} \end{cases} \tag{5}$$

Exploration softmax

L'exploration softmax (aussi appelée Max-Boltzmann) se base sur la fonction softmax qui permet de transformer un vecteur de réels en une distribution de probabilités sur les éléments du vecteurs. La fonction softmax est illustrée dans l'Équation 6, où V est un vecteur de réels.

$$\sigma_i(V) = \frac{e^{V_i}}{\sum_{j \in V} e^{V_j}} \tag{6}$$

Dans le cadre de l'apprentissage par renforcement, on utilise le vecteur $\{Q(s, a) \mid a \in A\}$ pour pondérer la probabilité de chaque action, et on ajoute un terme τ appelé « température » qui permet de contrôler le niveau d'exploration. L'Équation 7 illustre la probabilité de prendre l'action a dans l'état s avec une exploration softmax.

$$\pi_a(s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a' \in A} e^{\frac{Q(s,a')}{\tau}}} \tag{7}$$

Notez que quand $\tau \to \infty$, toutes les actions sont équiprobables (exploration maximale), et que lorsque $\tau \to 0$, la stratégie devient gloutonne (exploitation maximale) et choisit presque toujours l'action avec la plus haute valeur Q.

4. Rapport

On vous demande d'écrire un rapport concis et structuré sous la forme d'un texte suivi (par opposition à des réponses courtes sans contexte). Dans ce rapport, vous devez mener plusieurs expériences, expliquer votre mode opératoire, présenter vos résultats et en discuter. Sans mention contraire, utilisez $p = 0.1$ pour la valeur de non-déterminisme de l'environnement.

Note: Dans un rapport scientifique, il est très important que les expériences soient reproductibles. Par conséquent, pour tous les résultats que vous produisez, il est fondamental d'expliquer minutieusement votre mode opératoire et les paramètres utilisés. Ce n'est pas parce que les paramètres sont indiqués dans cet énoncé que vous ne devez pas les indiquer dans votre rapport !

4.1. Value Iteration

Pour l'algorithme de Value Iteration, on vous demande d'entraîner l'algorithme avec les valeurs de $\delta \in \{1, 0.1, 0.01, 0.005, 0.001\}$ et d'analyser

- le nombre d'itérations k nécessaires à converger;
- la valeur de chaque état à la fin de l'algorithme.

[1]Par exemple avec seaborn ou matplotlib.

Reportez dans une table le nombre d'itérations nécessaires pour atteindre une convergence. Ensuite, produisez une *heat map*[1] semblable à la Figure 2 qui représente les valeurs de chaque état à la fin de l'algorithme, et déduisez-en la stratégie optimale.

Ensuite, choisissez deux valeurs de p pour lesquelles la stratégie optimale est différente pour $\delta = 0.01$. Produisez deux *heat maps* pour le montrer.

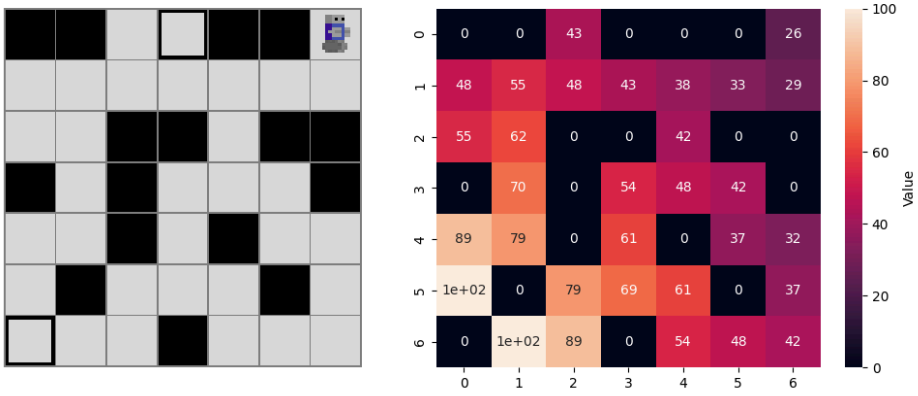

Figure 2. – Exemple de *heat map* obtenue pour un autre labyrinthe.

4.2. Q-learning

Pour l'algorithme du Q-learning, on vous demande d'analyser les performances de votre agent au cours de l'entraînement pour différentes méthodes d'exploration. Dans chaque cas, entraînez votre agent pendant minimum 20 000 interactions avec l'environnement.

Utilisez les paramètres suivants:

- ε-greedy avec $\varepsilon = 0$
- ε-greedy avec $\varepsilon = 0.1$
- ε-greedy avec $\varepsilon = 0.5$
- ε-greedy avec ε qui diminue linéairement de 1 à 0.01 au cours de l'entraînement
- softmax avec $\tau = 0.01$
- softmax avec $\tau = 1$
- softmax avec $\tau = 10$
- softmax avec τ qui diminue exponentiellement de 100 à 0.01 au cours de l'entraînement

On vous demande de produire un graphique sur lequel ces huit courbes d'entraînement (selon les différents paramètres) sont visibles. Chaque courbe d'entraînement indique le score[2] au cours de l'entraînement (de $t = 0$ à $t = 20000$ ou plus).

Notez que comme l'exploration est aléatoire, il est important de répéter votre expérience plusieurs fois pour vous assurer d'un comportement *moyen*. Dans le cadre de ce projet, on vous demande de répéter vos expériences minimum 10 fois avec des graines aléatoires (*seed*, voir `main.py`) différentes et de tracer la moyenne sur vos graphiques.

Note: Une partie de votre travail consiste à fixer certains paramètres (α, nombre d'étapes d'entraînement, γ, ...) et à déterminer comment représenter les données (quel traitement effectuer, quel type de graphique utiliser). Veillez à documenter votre méthodologie.

[2] La somme des récompenses au cours d'un épisode.

4.3. Discussion

- Comment l'aspect aléatoire de l'environnement affecte l'apprentissage? De quelle façon les méthodes de value iteration et q-learning gèrent-elles le non-déterminisme? En d'autre termes, l'aspect aléatoire de l'environnement est-il encodé dans les valeurs apprises? Si oui, comment?
- Comment les stratégies d'exploration ε-greedy et Max-Boltzmann influencent-elles l'équilibre entre exploration et exploitation ? Laquelle de ces stratégies semble atteindre un meilleur équilibre, et pourquoi ? Soutenez vos arguments à l'aide de vos résultats.
- Quel est l'effet du discount factor γ ? À partir de quelle valeur la sortie optimale change-t-elle? Justifiez votre réponse (expérimentalement ou formellement).
- Quel est l'effet du taux d'apprentissage α? Justifiez votre réponse à l'aide de vos résultats.
- Le cas échéant, expliquez votre utilisation des grands modèles de langage (ChatGPT, Claude, ...).

Remise

Le livrable de ce projet se présente sous la forme d'un fichier zip contenant vos sources python ainsi que votre rapport en PDF. Nous vous encourageons à utiliser un outil de rédaction scientifique tel que Typst ou Latex (par exemple avec Overleaf) pour rédiger votre rapport.

Ce travail est **individuel** et doit être rendu sur l'Université Virtuelle pour le 30/11/2025 à 23:59.

Veuillez adresser vos questions à Yannick Molinghen à l'adresse yannick.molinghen@ulb.be.