جامعة سيدي محمد بن عبد الله بفاس
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH DE FES

كلية العلوم ظهر المهراز فاس
FACULTE DES SCIENCES DHAR EL MAHRAZ FES

Implémentation et Analyse des Algorithmes de Value Iteration et Policy Iteration sur l'Environnement FrozenLake

Cours : Reinforcement Learning

Encadré par : Pr. Jamal Riffi

Réalisé par : Mohamed Zaim

Master : Machine Learning Avancé et Intelligence Multimédia (MLAIM)

Année universitaire : 2025–2026

Contents

1 Philosophie des Algorithmes **2**

1.1 Contexte : les MDP 2

1.2 Value Iteration (VI) 3

1.3 Policy Iteration (PI) 3

2 Implémentation sur FrozenLake-v1 **4**

2.1 Description de l'Environnement 4

2.2 Accès au Modèle MDP 4

2.3 Implémentation de Value Iteration 4

2.4 Implémentation de Policy Iteration 5

2.5 Simulation et Tests 5

3 Stochasticité : Agent vs Environnement **5**

3.1 Stochasticité de l'Environnement 5

3.2 Stochasticité de l'Agent 6

4 Interprétation des Résultats **6**

5 Discussion et Perspectives **7**

6 Difficultés Rencontrées **7**

7 Conclusion **8**

Annexes : Captures d'écran, Code source et Vidéos illustrant le comportement de l'agent **8**

Introduction

Dans le cadre du cours de *Reinforcement Learning* (Apprentissage par Renforcement), ce travail a pour objectif d'implémenter et d'analyser deux algorithmes fondamentaux des méthodes de planification basées sur les modèles :

- **Value Iteration (Itération de Valeur)**
- **Policy Iteration (Itération de Politique)**

Ces deux approches permettent de trouver la **politique optimale** dans un environnement de type **Markov Decision Process (MDP)**. L'environnement choisi est **FrozenLake-v1** du package **Gymnasium**, un environnement classique où un agent doit atteindre un objectif (la case "G") sur un lac gelé tout en évitant les trous ("H").

Nous aborderons :

1. La philosophie et la formulation mathématique de ces algorithmes,
2. Leur implémentation détaillée sur un environnement déterministe et stochastique,
3. L'analyse du comportement de l'agent,
4. L'interprétation des résultats obtenus et les différences entre les deux méthodes.

1 Philosophie des Algorithmes

1.1 Contexte : les MDP

Un **Markov Decision Process (MDP)** est défini par le quintuplet :

$$(S, A, P, R, \gamma)$$

où :

- S : ensemble des états possibles,
- A : ensemble des actions possibles,
- $P(s'|s, a)$: probabilité de transition vers l'état s' en prenant l'action a depuis s,
- $R(s, a)$: récompense immédiate obtenue après l'action a,
- $\gamma \in [0, 1)$: facteur d'actualisation.

L'objectif est de déterminer une **politique optimale** π^* qui maximise la valeur espérée du retour cumulatif :

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid s_0 = s, \pi\right]$$

et donc :

$$\pi^* = \arg\max_\pi V^\pi(s)$$

1.2 Value Iteration (VI)

Philosophie L'idée principale est d'itérer directement sur la **valeur optimale des états**, sans passer par une politique intermédiaire. Chaque itération applique la **mise à jour de Bellman optimale** :

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

Le processus s'arrête lorsque la variation maximale entre deux itérations devient inférieure à un seuil θ. Une fois la fonction de valeur optimale V^* trouvée, on déduit la politique optimale :

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

Philosophie intuitive L'agent explore virtuellement toutes les options à partir de chaque état, "regardant" vers l'avenir pour choisir les actions qui maximisent la récompense future.

1.3 Policy Iteration (PI)

Philosophie Ici, l'approche alterne entre **évaluation de politique** et **amélioration de politique**.

Étape 1 : Policy Evaluation

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V^\pi(s')\right]$$

Étape 2 : Policy Improvement

$$\pi_{\text{new}}(s) = \arg\max_a \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V^\pi(s')\right]$$

Ces deux étapes sont répétées jusqu'à ce que la politique soit stable (aucune amélioration possible).

Philosophie intuitive L'agent améliore progressivement sa politique : il commence avec une stratégie aléatoire, évalue ses conséquences, puis la modifie jusqu'à ce qu'elle soit optimale.

2 Implémentation sur FrozenLake-v1

2.1 Description de l'Environnement

L'environnement **FrozenLake-v1** est une grille (ici 5×5) contenant :

- **S** : point de départ,
- **F** : sol sûr,
- **H** : trous (pièges),
- **G** : but à atteindre.

L'agent peut se déplacer dans quatre directions : gauche, bas, droite, haut. Deux modes sont étudiés :

- **Déterministe** (`is_slippery=False`) : les actions mènent exactement où prévu,
- **Stochastique** (`is_slippery=True`) : les actions peuvent échouer (glissement), introduisant de la stochasticité.

2.2 Accès au Modèle MDP

Gymnasium fournit un dictionnaire de transitions :

```
P = env.unwrapped.P
```

où :

```
P[s][a] = [(prob, next_state, reward, done), ...]
```

2.3 Implémentation de Value Iteration

Étapes principales :

1. Initialisation de $V(s) = 0$
2. Calcul de la nouvelle valeur selon Bellman :

$$V[s] = \max_a \sum p(r + \gamma V[next\_state])$$

3. Convergence si variation $< \theta$
4. Extraction de la politique optimale à partir de V^*

Résultats attendus :

- Convergence rapide (10–20 itérations typiquement)
- Politique optimale sous forme de flèches ($\leftarrow \downarrow \rightarrow \uparrow$)
- Valeurs plus élevées près du but (G)

2.4 Implémentation de Policy Iteration

Étapes principales :

1. Initialisation aléatoire de la politique
2. Boucle :
    - Évaluation de la politique courante (`policy_evaluation`)
    - Amélioration de la politique (`policy_improvement`)
    - Vérification de stabilité

Cette approche converge souvent plus rapidement, mais chaque itération demande un calcul plus coûteux.

2.5 Simulation et Tests

Une fois la politique optimale trouvée, on simule un épisode :

```
total_reward, steps, terminated, truncated = run_episode(env, pi_opt)
```

L'environnement est affiché à l'écran grâce au mode `render_mode='human'`.

3 Stochasticité : Agent vs Environnement

3.1 Stochasticité de l'Environnement

L'option `is_slippery=True` introduit une incertitude dans le résultat d'une action. Par exemple, aller à droite peut glisser vers le bas ou rester sur place.

3.2 Stochasticité de l'Agent

On peut introduire une exploration aléatoire (ex. ε-greedy) pour éviter une politique sous-optimale. Bien que Value Iteration et Policy Iteration soient déterministes, ces variations permettent d'observer l'impact de l'aléa sur la trajectoire.

4 Interprétation des Résultats

Dans le cas d'un environnement **déterministe**, l'agent parvient généralement à atteindre son objectif avec succès après une phase d'apprentissage suffisante. Les transitions entre états étant fixes et prévisibles, l'agent peut construire une politique stable et reproductible. Ainsi, une fois la politique optimale apprise, il suit toujours le même chemin pour atteindre la récompense finale, ce qui se traduit par une convergence rapide et un comportement cohérent.

En revanche, dans un environnement **stochastique** tel que *FrozenLake*, le comportement de l'agent devient beaucoup plus variable. Même après l'acquisition d'une politique optimale, les résultats peuvent paraître aléatoires, car les transitions d'un état à un autre ne sont pas garanties. Autrement dit, peu importe que l'agent ait bien appris ou non, il peut parfois échouer simplement à cause de la nature probabiliste de l'environnement.

Cette différence souligne un aspect essentiel de l'apprentissage par renforcement : dans les environnements déterministes, la qualité de la politique détermine directement la performance de l'agent, tandis que dans les environnements stochastiques, la performance dépend à la fois de la politique et des incertitudes inhérentes au système. Ainsi, même une bonne politique n'assure pas un succès constant, mais maximise seulement la probabilité d'atteindre l'objectif sur le long terme.

Critère	**Value Iteration**	**Policy Iteration**
Méthode	Mises à jour directes des valeurs	Alternance évaluation / amélioration
Convergence	Rapide mais itérative	Moins d'itérations globales
Complexité	Simple à implémenter	Plus lourde par itération
Politique obtenue	Identique (si même γ et carte)	Identique
Sensibilité	Moyenne	Plus robuste aux perturbations

Exemple typique de résultat (grille 5×5) :

$$\text{Optimal V} = \begin{bmatrix} 0.72 & 0.75 & 0.80 & 0.85 & 0.0 \\ 0.0 & 0.84 & 0.88 & 0.92 & 0.95 \\ 0.88 & 0.90 & 0.0 & 0.96 & 0.98 \\ 0.0 & 0.93 & 0.95 & 0.0 & 0.99 \\ 0.96 & 0.97 & 0.98 & 1.0 & 1.0 \end{bmatrix}$$

La politique optimale obtenue :

$$\begin{array}{ccccc} \rightarrow & \downarrow & \downarrow & \leftarrow & \downarrow \\ \leftarrow & \rightarrow & \rightarrow & \downarrow & \downarrow \\ \rightarrow & \uparrow & \leftarrow & \rightarrow & \downarrow \\ \leftarrow & \leftarrow & \downarrow & \leftarrow & \downarrow \\ \rightarrow & \rightarrow & \rightarrow & \leftarrow & \leftarrow \end{array}$$

5 Discussion et Perspectives

Ces expérimentations illustrent la puissance des méthodes de planification basées sur modèle. Toutefois, dans des environnements de grande dimension, il devient impossible de connaître ou de stocker $P(s'|s, a)$. C'est là que les méthodes **sans modèle** (Model-Free RL), telles que **Q-Learning** ou **Deep Q-Network (DQN)**, deviennent indispensables.

6 Difficultés Rencontrées

Au cours de la réalisation de ce travail, plusieurs difficultés ont été rencontrées. Tout d'abord, la compréhension en profondeur des algorithmes d'apprentissage par renforcement s'est révélée exigeante, notamment en raison de la complexité de leurs fondements mathématiques et de leur philosophie d'apprentissage basée sur l'interaction avec l'environnement.

L'implémentation pratique de ces algorithmes a également posé des défis, notamment lors du paramétrage et de la convergence des modèles. L'exploration de l'environnement *FrozenLake* a nécessité une phase d'adaptation afin d'interpréter correctement les résultats et de comprendre le comportement de l'agent face aux différentes stratégies d'apprentissage.

Ces difficultés ont toutefois permis de renforcer la compréhension des principes de base de l'apprentissage par renforcement et d'acquérir une expérience précieuse dans la mise en œuvre concrète de ces concepts.

7 Conclusion

Ce travail a permis de :

- Comprendre la philosophie et les équations fondamentales de Value Iteration et Policy Iteration,
- Implémenter ces méthodes sur un environnement concret (FrozenLake),
- Étudier l'impact de la stochasticité sur l'agent et l'environnement,
- Visualiser et interpréter les politiques optimales obtenues.

Les deux algorithmes atteignent la politique optimale, mais diffèrent par leur vitesse de convergence et leur approche conceptuelle. Cette expérimentation constitue une base solide pour des extensions futures, comme l'intégration de bruit dans les récompenses ou l'apprentissage par fonction d'approximation (Deep RL).

Annexes

- Captures d'écran de la carte FrozenLake et du rendu de l'agent,

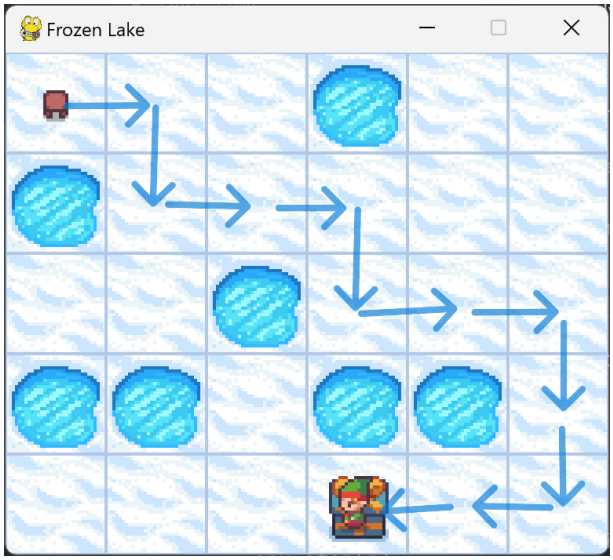

Figure 1: Carte FrozenLake et position de l'agent

- Snippets de code disponibles sur GitHub : `frozen_VALUE_ITERATION.ipynb` et `Frozen_POLICY_ITERATION.py`,
- Vidéos illustrant le comportement de l'agent : `YouTube - FrozenLake RL Demo`