

## **Introduction à l'apprentissage automatique**

### **Rapport projet n°2 - Equation de Bellman et Q-learning/SARSA**

*Répartition du travail : Sébastien Patte pour la première partie, Gwenn Renouard pour la deuxième, avec tests et mise au point en commun.*

#### **Première partie : Equation de Bellman**

##### **Introduction :**

Le projet concerne l'apprentissage par renforcement. On considère un monde de type Gridworld dans lequel un agent peut se déplacer (actions) avec des cases comportant des murs ou des récompenses. Il s'agit d'implémenter en Java des fonctionnalités de calculs de la fonction de valeur  $V$  (par deux méthodes différentes, inversion de matrice et itération), et d'amélioration de la politique jusqu'à obtenir une politique d'action satisfaisante, puis de comparer les résultats.

Le sujet comporte deux grilles permettant de tester les fonctionnalités développées.

##### **Représentation de la grille :**

Ci-dessous une représentation de la grille donnant la correspondance entre le système de coordonnées et les états (exemple pour une grille de largeur 5 et de hauteur 5).

Chaque case contient son numéro d'état et ses coordonnées dans la grille sous la forme '(i,j)', par exemple, l'état n°7 correspond à la case  $i=2$  et  $j=1$ .

0 (0,0)	1 (1,0)	2 (2,0)	3 (3,0)	4 (4,0)	→ i
5 (0,1)	6 (1,1)	7 (2,1)	8 (3,1)	9 (4,1)	
10 (0,2)	11 (1,2)	12 (2,2)	13 (3,2)	14 (4,2)	
15 (0,3)	16 (1,3)	17 (2,3)	18 (3,3)	19 (4,3)	
20 (0,4)	21 (1,4)	22 (2,4)	23 (3,4)	24 (4,4)	
↓ j					

Cette représentation de la grille, avec le  $j$  vers le bas, est utilisée par toutes les fonctions d'affichage/écriture du projet (certaines fonctions du squelette ont dû être modifiées pour y correspondre).

Par ailleurs, le déplacement vers le haut qui faisait  $j+=1$  a été modifié en  $j-=1$  pour qu'un déplacement vers le haut corresponde vraiment à un déplacement vers le haut sur la grille qu'on affiche. (Et réciproquement pour le déplacement 'bas' qui passe de  $j-=1$  à  $j+=1$ ).

##### **Algorithmes des deux méthodes :**

###### **1) calcul de $V$ par inversion de matrice :**

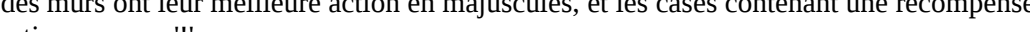
On prend une politique initiale uniforme (chaque action est équiprobable).

On initialise  $V$  avec la politique initiale.

Pour chaque itération :

On améliore la politique avec `improvePolicy()` à partir de  $V$ .

On calcule  $V$  à partir de l'inversion de la matrice  $P^\pi$  et le vecteur de récompense moyenne 'this.reward'.

- On écrit dans les fichiers 'Politique/Inversion.txt' et 'Politique/Iteration.txt' les valeurs de la politique pour chaque mode de calcul.
- 'Inversion.txt' contient la politique de chaque itération pour le calcul de V par inversion de matrice. Et 'Iteration.txt' contient la même chose pour le calcul de V par itération.
- On a donc l'évolution de la politique représentée par la grille des meilleures actions possibles où les cases contenant des murs ont leur meilleure action en majuscules, et les cases contenant une récompense ont leur meilleure action avec un '!'.  


## Interprétation des résultats :

Pour toutes les grilles, les résultats de V et de la politique 'this.action' sont les mêmes avec les deux méthodes. A chaque itération la politique et les valeurs de V s'étendent sur la grille en partant de la récompense. Les deux méthodes semblent avoir la même évolution de la politique et de V (pour chaque itération), mais en observant les temps d'exécution on observe que la méthode par inversion de matrice prend en moyenne plus lente que la méthode par itération pour la même grille.

Le temps d'exécution de chaque méthode est affiché en fin d'affichage console.

## Fonctionnalité supplémentaire mise en place : mouvement 'saut' et nouvelle grille

Le mouvement 'saut' est un mouvement ajouté qui déplace l'agent de 2 cases vers la gauche.

Il peut être activé/désactivé par le second paramètre du constructeur 'GridWorld\_sql'.

Ce second paramètre est un booléen qui active ou désactive le mouvement 'saut'.

Dans createGrid() on crée une nouvelle grille, la grille '2'.

On peut donc choisir la grille '2' avec le premier paramètre du constructeur 'GridWorld\_sql'.

La grille '2' est construite de la sorte :

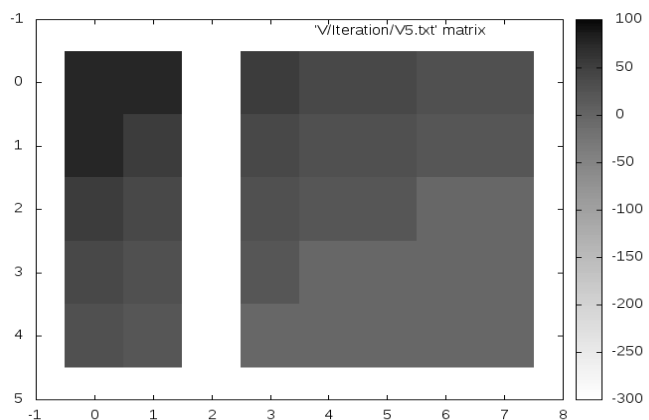
Grille des récompenses :

```
20.0 -1.0 -1000.0 -1.0 -1.0 -1.0 -1.0 -1.0
-1.0 -1.0 -1000.0 -1.0 -1.0 -1.0 -1.0 -1.0
-1.0 -1.0 -1000.0 -1.0 -1.0 -1.0 -1.0 -1.0
-1.0 -1.0 -1000.0 -1.0 -1.0 -1.0 -1.0 -1.0
-1.0 -1.0 -1000.0 -1.0 -1.0 -1.0 -1.0 -1.0
```

Les récompenses '-1000,0' sont des murs.

*Cette grille est utile pour tester le mouvement 'saut' car on est obligé de 'sauter' par dessus les murs de la troisième colonne pour accéder à la récompense (en (0,0)) si on commence du côté droit de la grille.*

*Ci-contre à droite, l'image correspondant aux valeurs de V pour l'itération n°5, où l'on peut constater que le saut est bien pris en compte, le dégradé s'étendant au-delà du mur blanc situé en troisième colonne.*



## Deuxième partie : Q-learning/SARSA pour pacman

### Introduction :

A partir d'un programme simulant un jeu ressemblant à pacman, dans lequel Pacman se déplace aléatoirement et peut rencontrer soit un fantôme soit de la nourriture, il s'agit de développer une fonctionnalité d'apprentissage par renforcement (en utilisant les deux méthodes Q-learning et SARSA) et l'utiliser pour les prochains mouvements.

A chaque itération du monde, la fonction d'apprentissage (fonction 'learn') met à jour la fonction de valeur Q(s,a) et pacman choisit une action (fonction 'chooseAction').

### **Algorithmes des deux méthodes :**

La fonction d'apprentissage met à jour le  $Q(s,a)$  en utilisant une des deux méthodes suivantes.

#### **1) Q-learning :**

A partir de l'état  $s'$ , on regarde la meilleure action en prenant le maximum de la valeur de  $Q(s',a')$  en faisant varier  $a'$ .

Puis on met à jour  $Q(s,a)$  selon la formule  $Q(s,a) = Q(s,a) + \alpha(\text{reward} + \gamma \cdot Q_{\max}(s',a') - Q(s,a))$

#### **2) SARSA :**

A partir de l'état  $s'$ , on choisit  $a'$  également selon les valeurs de  $Q(s',a')$ , mais de manière e-greedy.

Puis on met à jour  $Q(s,a)$  selon la formule  $Q(s,a) = Q(s,a) + \alpha(\text{reward} + \gamma \cdot Q(s',a') - Q(s,a))$

### **Interprétation des résultats :**

#### **1) Déplacement aléatoire :**

#### **2) Q-learning :**

#### **3) SARSA :**