Projet : PROCESSUS DE DÉCISION MARKOVIEN

Un cas particulier : Homer VS Donuts

AARAB MARYAM
BENELKATER MOHAMED
CORRIOU ALEXANDRE
PREVOT ALEXIA

Encadrant :
Jeanne BARTHÉLEMY
MAIN3
2021/2022

20 mai 2022

Table des matières

Introduction

1 Présentation du sujet : cas Homer Vs Donuts

Le sujet de ce projet est de réaliser un jeu. Il consiste à aider Homer à trouver le chemin le plus court vers les donuts en évitant les ennemis, c'est-à-dire marquer le plus gros score sachant que des cases permettent de gagner ou perdre des points. Les règles de ce jeu sont les suivantes :

- l'agent est dans une grille ;
- l'agent ne peut pas sortir de la grille ;
- l'agent ne se déplace pas comme il le souhaite (il a une chance de se tromper) ;
- l'agent reçoit une récompense à chaque déplacement en fonction de la case où il atterri (case vide : SCORE - 1, case avec ennemi : SCORE - 100, case avec donuts : SCORE + 100) ;
- lorsque l'agent atteint l'objectif (donuts) ou un ennemi (Bart, Marge,...), le jeu se termine et il revient à la position de départ.

Dans ce projet nous devrons donc réaliser un code visant à trouver le chemin le plus court vers les donuts tout en évitant les ennemis. C'est-à-dire nous devrons créer un environnement, des probabilités de se tromper, des actions et des récompenses en fonction de la position dans l'environnement. Pour cela, nous devrons réaliser un modèle traduisant les paramètres de ce jeu.

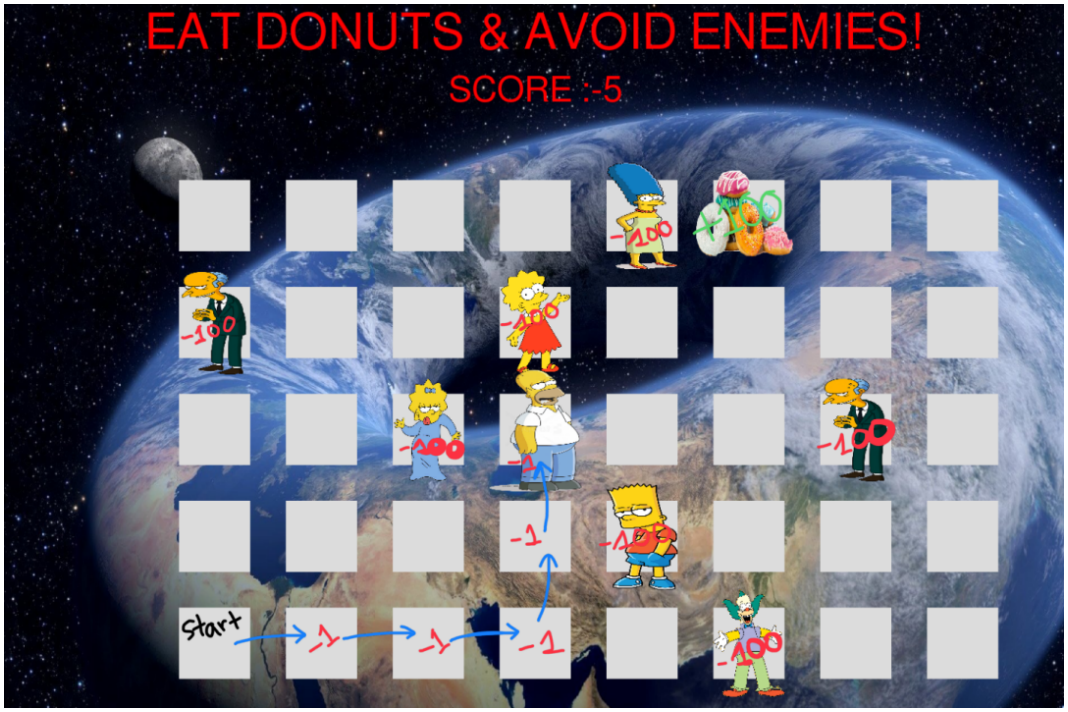


FIGURE 1 – Exemple de l'interface à obtenir

2 Processus décisionnel Markovien (MDP)

Dans notre projet, l'objectif sera d'entraîner Homer grâce à un modèle afin qu'à l'issue de l'entraînement, il trouve le plus court chemin jusqu'aux donuts et ainsi que le problème soit résolu.

Il existe différents types d'apprentissage.
Mais tout d'abord il faut modéliser le problème et pour cela nous allons utiliser un processus décisionnel markovien (MDP). Un MDP est un quadruplet { S , A , T , R } qui définit :

- **S** : qui représente l'ensemble d'états que l'environnement peut prendre ;
- **A** : l'ensemble des actions applicable dans notre environnement à savoir les déplacements de l'agent d'une case à l'autre ;
- **T** : la fonction probabilité de transition qui définis la probabilité que l'agent passe d'un état à un autre en fonction de l'action appliqué ;
- **R** : la fonction qui associe chaque état de l'environnement à une récompense.

Le but de ce MDP est d'apprendre des stratégies d'action qui maximisent les gains.[1][5]

3 Démarche envisagée

Ainsi l'application d'un processus décisionnel markovien semble être la voie pour résoudre notre problème. On va donc se pencher sur la définition de notre processus décisionnel markovien.

Première partie

MDP : cas d'Homer Vs Donuts et apprentissage pour faire le meilleur score

1 Détails du MDP

Nous allons détailler dans cette partie notre processus de décision de Markov.

Le processus décisionnel markovien (MDP) est défini par l'ensemble de 4 termes suivants :

- Ensemble d'états : S ;
- Ensemble d'actions : A ;
- Fonction de transition : T pour tout tuple (s, a, s') ;
- Fonction de récompense : R.

Appliquons chacun des termes aux paramètres du problème expliqué précédemment.

1.1 États et actions dans l'environnement

L'ensemble d'états représente l'ensemble d'états que l'environnement peut prendre, et dans ce cas, il représente toutes les cellules de la grille.

Un ensemble d'actions est un type d'action qu'un agent peut entreprendre. Dans ce cas, il existe quatre types d'actions car Homer peut se déplacer vers le haut, le bas, la gauche et la droite. On exclut les diagonales ou les sauts de cases.

Par exemple, dans la figure 2 c'est une grille 3x3. Concernant le déplacement, les quatres façons possibles de se déplacer sont représentées sur la figure par des flèches directionnelles ($\uparrow, \downarrow, \leftarrow, \rightarrow$).

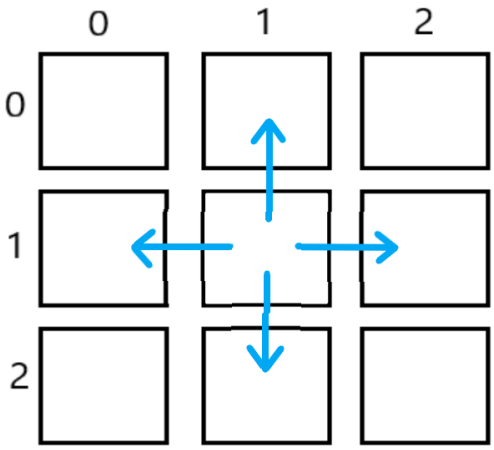


FIGURE 2 – Grille 3x3

1.2 Probabilité de transition

La fonction de transition est la probabilité de transition vers l'état suivant s' lorsque l'action a est effectuée dans un état s.

D'abord lorsque l'agent est sur une case, on doit choisir où il ira ensuite soit au hasard entre les quatre cases qui l'entourent, soit avec l'objectif de gagner et de trouver le chemin vers les donuts. De plus, pour représenter une certaine part d'aléatoire, on suppose que Homer a une certaine probabilité de se tromper de déplacement. Ainsi en partant de la case (1,1) et qu'on veut qu'il aille à droite à la case (1,2) (voir figure 3), on prend en compte des probabilités (p_d,p_h,p_g,p_b) sur la case (1,1) pour le déplacement final d'Homer (voir figure 4). On a que la somme des quatre probabilités vaut 1. On calcule les probabilités de manière aléatoire mais aussi de sorte que dans le cas où la décision est d'aller à droite par exemple, $p_d > 0.5$. En effet, il faut quand même que la probabilité d'aller à l'endroit choisi soit assez élevée sinon les déplacements sont beaucoup trop aléatoires et ne permettent pas de trouver le meilleur chemin.

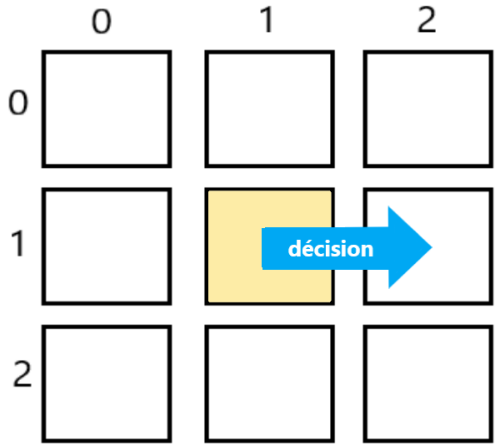


FIGURE 3 – Partant de (1,1) on décide d'aller à (1,2)

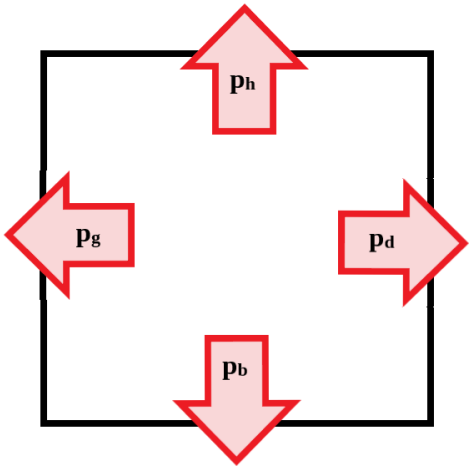


FIGURE 4 – Probabilité de transition pour la décision

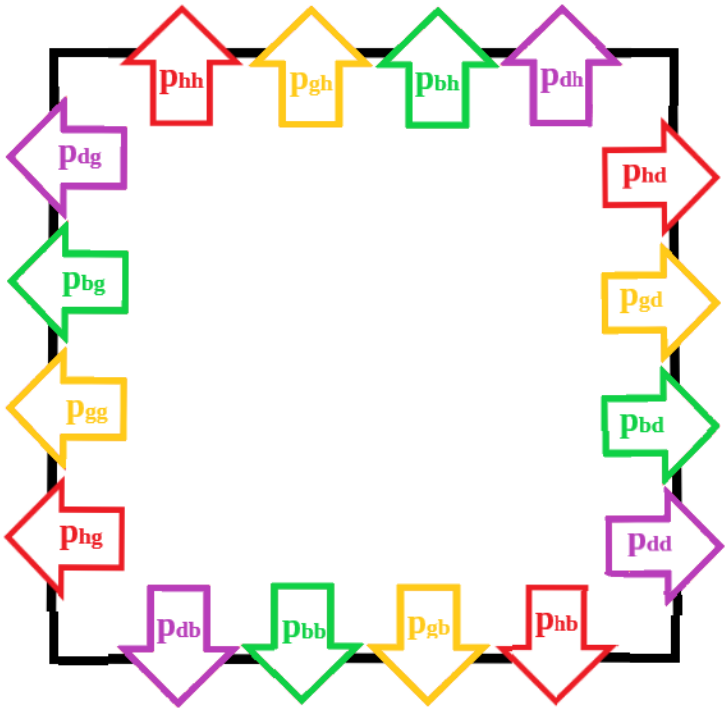


FIGURE 5 – Probabilité de transition pour la décision

Ainsi, pour chaque case il faut calculer seize probabilités différentes : quatres pour le choix d'aller à droite (p_{dd},p_{dg},p_{dh},p_{db}), quatres pour le choix d'aller à gauches (p_{gd},p_{gg},p_{gh},p_{gb}), etc. (voir figure 5) On a que les probabilités p_{dd}, p_{gg}, p_{hh} et p_{bb} sont supérieures à 0.5 et que la somme des probabilités pour une décision vaut 1.

1.3 Récompense

La récompense est la valeur reçue de l'environnement à la suite d'une nouvelle transition. Homer reçoit une récompense à chaque déplacement en fonction de la case où il atterri (case vide : SCORE - 1, case avec ennemi : SCORE - 100, case avec donuts : SCORE + 100). Cette récompense servira par la suite à aider pour choisir le meilleur chemin afin de maximiser le score final.

2 Apprentissage : Q-Learning

2.1 Q-Learning

Dans une simulation dite « model free », l'agent évolue à l'aveugle et ne connaît pas la récompense d'une action que quand il l'effectue. Pour trouver le chemin optimal, il doit donc effectuer un grand nombre de répétitions de l'expérience avant d'avoir des données suffisantes pour pouvoir savoir où aller. Ce genre de modèles n'est régi que par l'aléatoire.

Dans le cas de notre problème nous avons décidé d'utiliser un model-free qui est le Q-learning. Ce modèle ne nécessite aucun modèle initial de l'environnement. Ainsi cette méthode permet d'apprendre une stratégie sur un environnement non connu au départ.

La lettre Q désigne la fonction qui permet de calculer une valeur d'un état-action qui permet de calculer le gain potentiel : la récompense sur le long terme apportée, $Q[s, a]$ par le choix d'une action a dans un certain état s en suivant une politique optimale. La fonction $Q[s, a]$ est calculé à partir de l'équation suivante

$$Q^{new}[s_t, a_t] \longleftarrow Q[s_t, a_t] + \alpha.(rt + \gamma.maxQ(s_{t+1}, a) - Q[s_t, a_t]) \quad (1)$$

avec r_t la récompense reçu en passant de l'état s_t à l'état s_{t+1}, α un taux d'apprentissage compris entre 0 et 1 et γ un facteur d'actualisation.

Un taux d'apprentissage à 0 ne fait rien apprendre à l'agent alors qu'un taux à 1 fait que l'agent ne considère que les informations les plus récentes. Il est donc important de bien le choisir de manière à apprendre un peu à chaque transition mais sans pour autant changer de trop et rendre l'apprentissage compliqué. Le facteur d'actualisation permet de déterminer l'importance des récompenses futurs. Un facteur de 0 rend l'agent "myope" c'est à dire qu'il ne considère que la récompense actuelle tandis qu'un facteur de 1 fera tendre vers une récompense élevée à long terme. On peut remarquer que cela obéit bien à un chaîne de Markov étant donné que ça dépend uniquement de l'état actuel ainsi que des probabilités.[6]

Ainsi, grâce aux valeurs q calculées, on peut choisir à chaque case la prochaine à aller en prenant la valeur la plus élevée. Pour mieux comprendre le principe, la figure 6 ci-dessous représente un exemple dans la case d'une grille 3x3 et où Homer a effectué un grand nombre de répétitions de l'expérience et ayant collectés des données suffisantes pour trouver le chemin optimal.

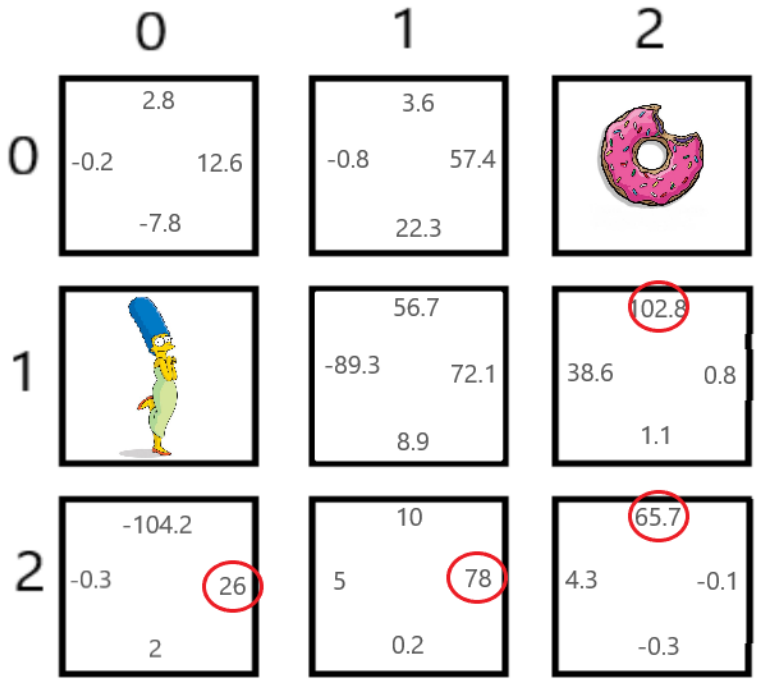


FIGURE 6 – Exemple de chemin optimal avec Q-learning

2.2 Exploration et exploitation

L'exploration est l'acte d'explorer l'environnement pour trouver des informations à son sujet. C'est plus un concept de bénéfice à long terme où elle permet à l'agent d'améliorer ses connaissances sur chaque action qui pourrait conduire à un bénéfice à long terme.

L'exploitation est l'acte d'exploiter les informations déjà connues sur l'environnement afin de maximiser le rendement. En exploitant essentiellement la valeur estimée actuelle de l'agent et il choisit l'approche gourmande pour obtenir le plus de récompenses. Cependant, l'agent est gourmand avec la valeur estimée et non avec la valeur réelle, il est donc probable qu'il n'obtienne pas le plus de récompenses.

Le dilemme de savoir s'il faut exploiter la connaissance partielle pour recevoir des récompenses ou s'il doit explorer des actions inconnues qui pourraient entraîner des récompenses beaucoup plus importantes.

Deuxième partie

Présentation du jeu et des paramètres

1 Les différentes fonctions d'apprentissages

Dans nos codes, nous avons choisi d'implémenter plusieurs fonctions d'apprentissage qui nous permettent de faire différents tests.

- play_to_learn_step permet d'apprendre une case par une case tout en gardant un historique des positions . Cette fonction permet d' "apprendre" à la fin d'un trajet (réussite ou échec de l'agent) ;
- play_to_learn_step2 permet d' "apprendre" une case par une case, tout en gardant les mêmes méthodes de mouvement que la fonction précédente ;
- play_to_learn effectue un trajet entier sans avoir besoin d'interagir avec l'utilisateur à chaque mouvement, tout en "apprenant" à chaque fin de trajet. play_to_win exploite les résultats obtenus précédemment en influant le choix des actions avec les Q-values calculées.

2 Taux d'apprentissage

Si l'on regarde la formule utilisée pour le Q-learning, on remarque que la nouvelle Q-valeur va être d'autant plus impactée et changée par le nouvel essai que le taux d'apprentissage est élevé. En effet, il détermine dans quelle mesure les nouvelles informations remplaceront les anciennes informations. Ainsi, un facteur élevé signifierait que l'agent s'intéressera uniquement aux dernières informations et au contraire un facteur faible rendrait l'apprentissage très lent vu qu'on ne modifie presque pas les valeurs.

2.1 Limites du taux d'apprentissage

2.1.1 Cas d'un taux élevé

On regarde les conséquences d'un taux d'apprentissage assez élevé, c'est-à-dire se rapprochant de 1 pris en paramètre au début du jeu.

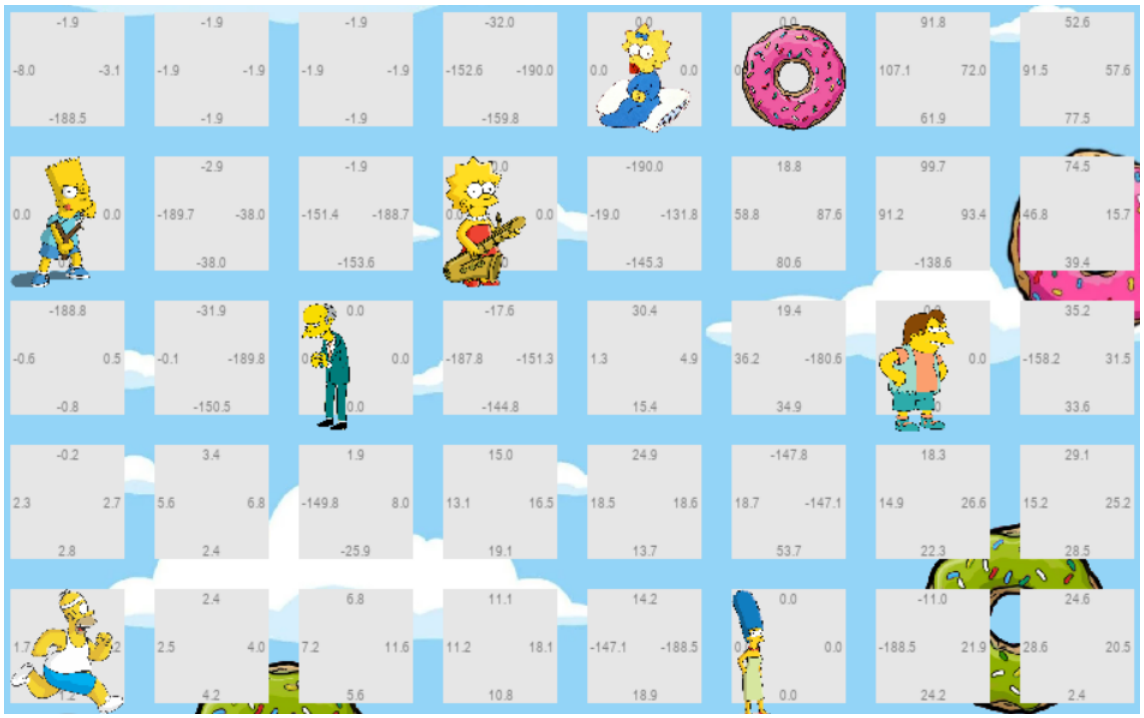


FIGURE 7 – alpha = 0.8 à t

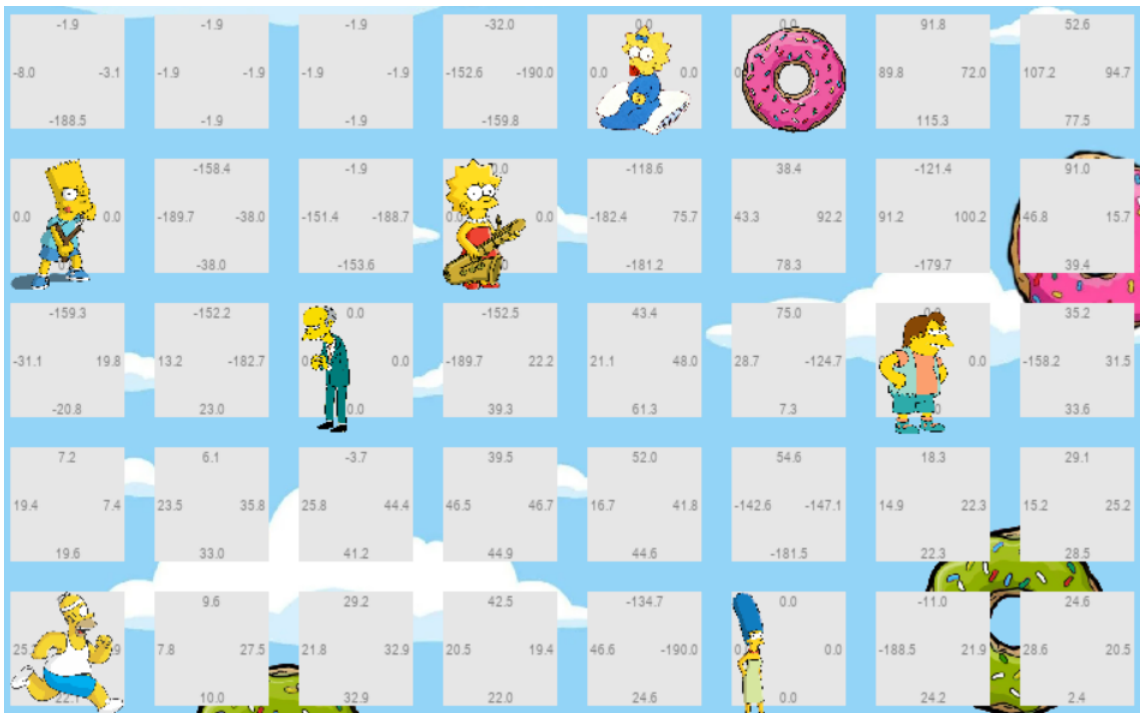


FIGURE 8 – alpha = 0.8 à t + δt

On remarque qu'il n'y a pas de réel chemin de trouvé pour atteindre les donuts par Homer, peu importe le temps passé à apprendre.

2.1.2 Cas d'un taux faible

On regarde les conséquences d'un taux d'apprentissage assez faible, c'est-à-dire se rapprochant de 0 pris en paramètre au début du jeu.

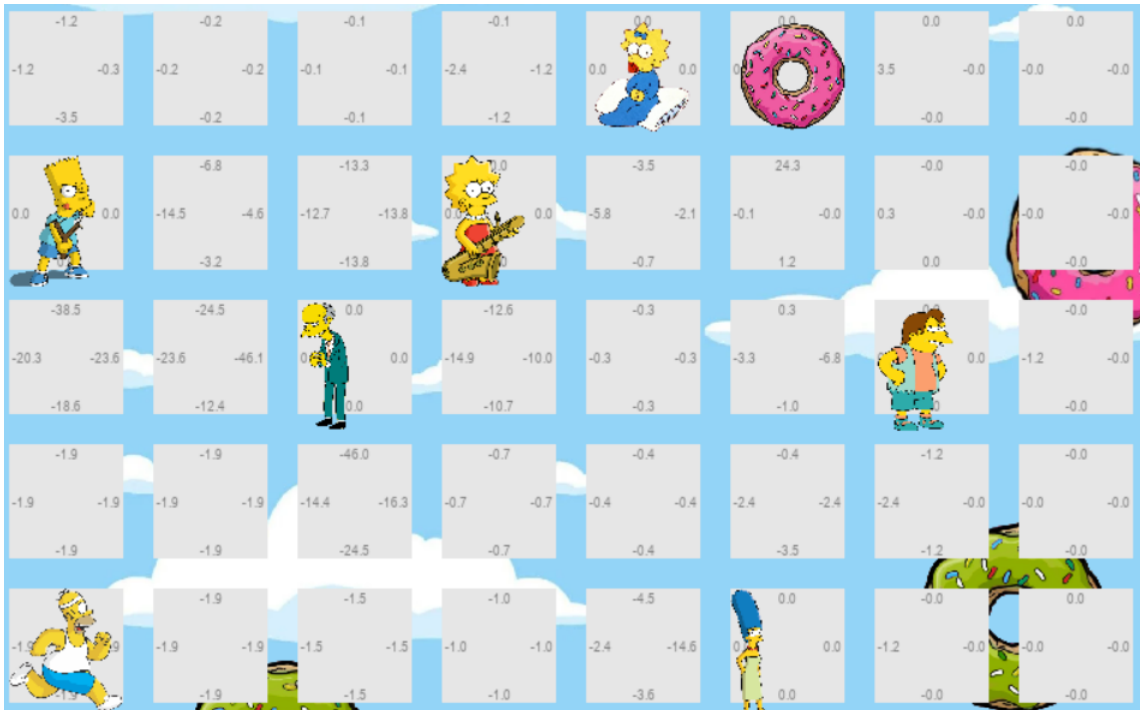


FIGURE 9 – alpha = 0.8 à t

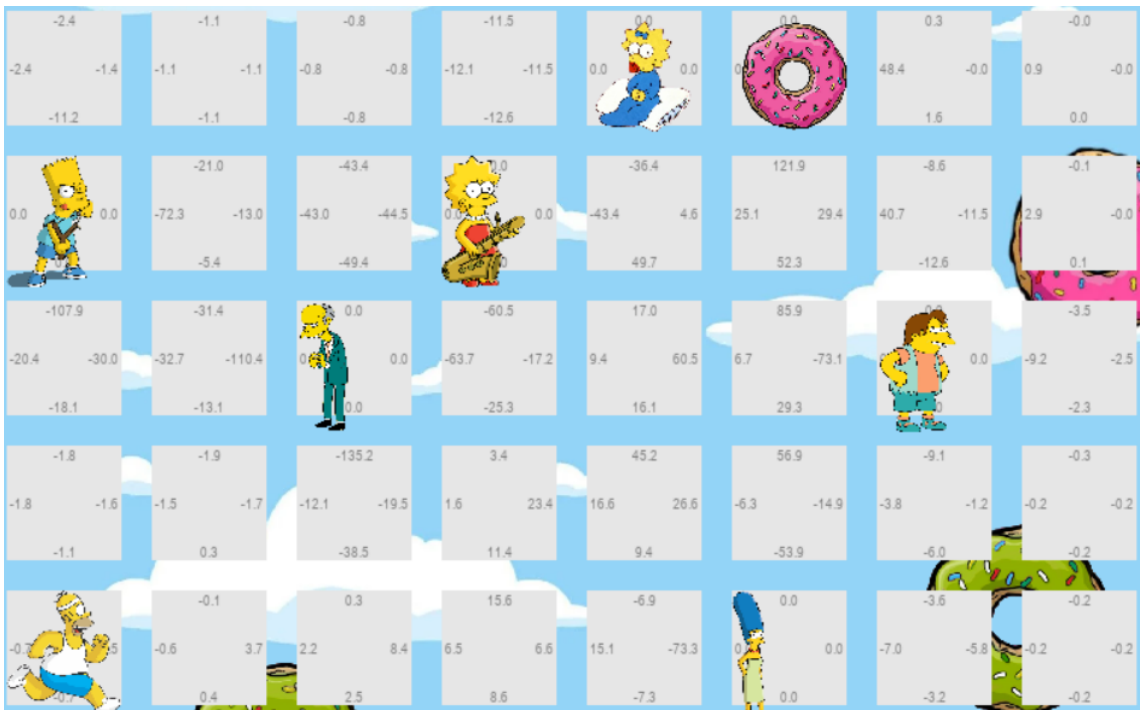


FIGURE 10 – alpha = 0.8 après très longtemps

On remarque qu'un chemin est effectivement trouvé et est très précis mais cependant il faut beaucoup plus de temps pour y parvenir.

Les observations de ces deux cas corroborent les hypothèses initiales puisque lorsque le taux d'apprentissage est trop élevé, le chemin ne se stabilise pas assez car il suffit d'un entraînement qui marche pas et toutes les valeurs sont grandement affectées. On observe alors les valeurs osciller de beaucoup. Et dans l'autre cas, lorsque le taux d'apprentissage est très faible, certes on finit par obtenir quelque chose d'intéressant mais c'est avec une durée très grande.

2.2 Variation du taux d'apprentissage à partir d'un certain moment

On commence par prendre un taux d'apprentissage cohérent qui permet de trouver un chemin au bout d'un certain temps. On le prend à 0.1. Après avoir appuyer sur la touche l et attendu un certain moment avant d'appuyer sur la touche s de manière à laisser un grand nombre de parties pour apprendre, on obtient le résultat ci-dessous sur la figure 7.

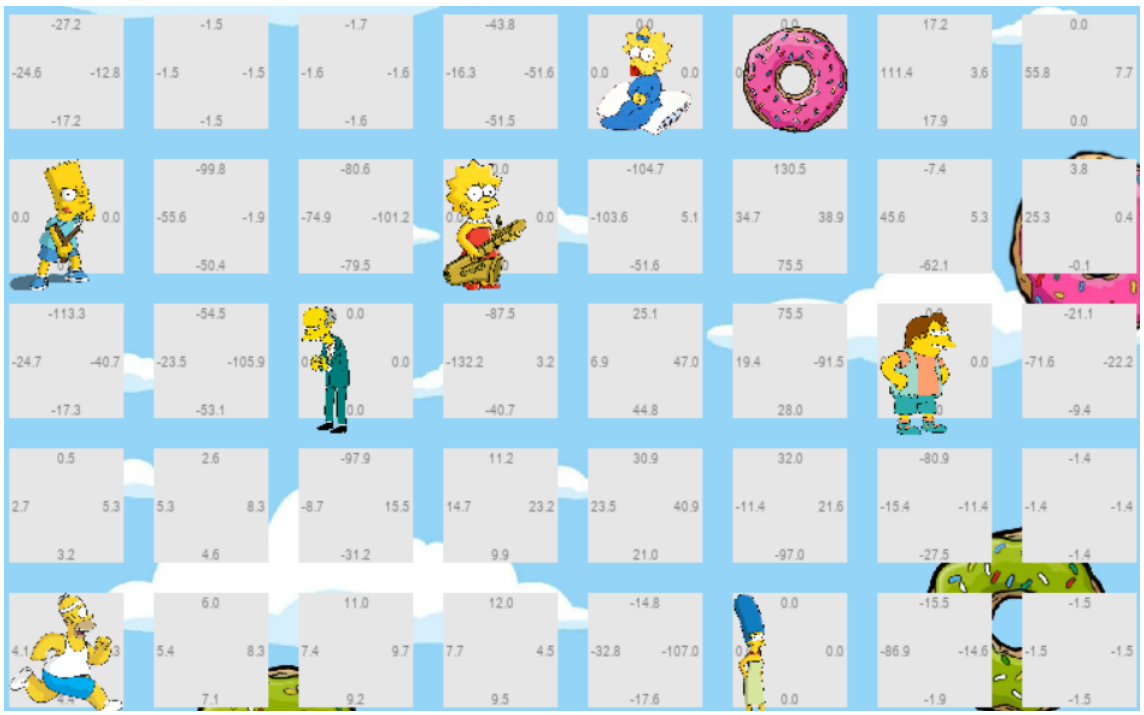


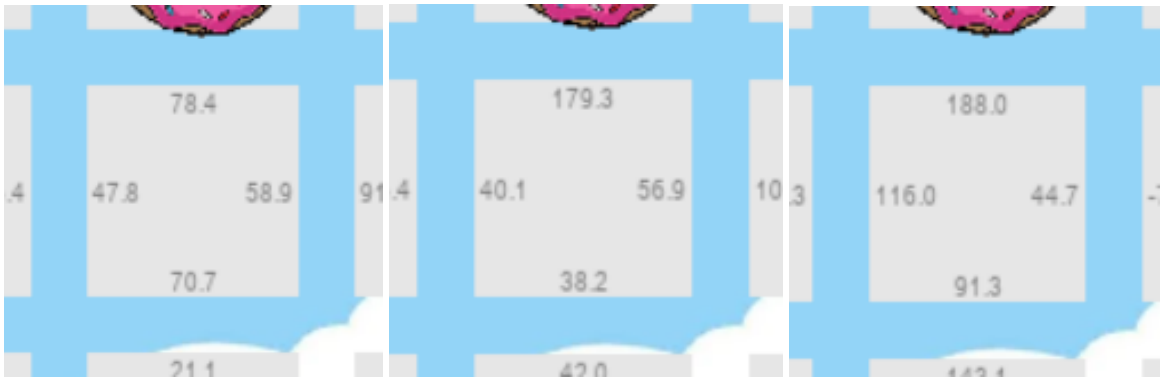
FIGURE 11 – Taux d'apprentissage de 0.1

Puis on va changer ce taux en cours d'apprentissage avec les touches * et / pour le réduire et l'augmenter.

Afin de faciliter les explications, on se concentre uniquement sur une seule case, par exemple celle à la 2-ème ligne et 6-ème colonne. On est à l'instant t et on regarde le résultat après quelques parties de plus (temps t + δt) et encore un deuxième fois (temps t + δt + Δt).

2.2.1 Augmentation du taux

A partir du taux à 0.1, à partir d'un moment on l'augmente à 0.4.



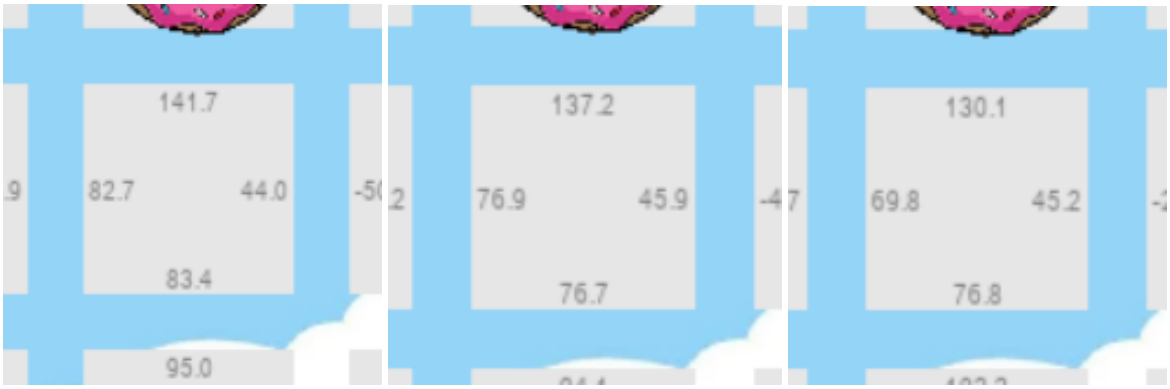
(a) (b) (c)

FIGURE 12 – alpha = 0.4 (a) à t (b) à t + δt (c) à t + δt + Δt

On observe ainsi des grandes différences entre les valeurs (passer de 78.4 à 179.3 à 188.0 par exemple sur la valeur du dessus) alors que l'apprentissage dure déjà depuis un bon moment. Ce résultat s'étend sur toute la grille.

2.2.2 Diminution du taux

On procède de la même manière en diminuant le taux d'apprentissage à 0.05.



(a) (b) (c)

FIGURE 13 – alpha = 0.05 (a) à t (b) à t + δt (c) à t + δt + Δt

On observe ainsi une certaine constance entre les valeurs (passer de 141.7 à 137.2 à 130.1 par exemple sur la valeur du dessus) une fois que le chemin final est à peu près trouvé. En effet, les valeurs ne s'éloignent pas trop et restent dans un intervalle. Ce résultat s'étend sur toute la grille.

Pour conclure sur le taux d'apprentissage, celui-ci est très important et doit être bien choisi afin de résoudre le problème posé. Il ne doit pas être trop grand sinon on observe de grands écarts entre deux parties pour apprendre et ainsi le chemin final ne se stabilise pas vraiment. Et, il ne faut pas non plus qu'il soit trop petit car le temps pour apprendre est beaucoup trop long pour un même résultat obtenu avec un temps plus raisonnable d'apprentissage. Ainsi, au début on peut prendre un taux relativement grand mais pas trop afin d'apprendre vite au début puis au fils du temps le réduire pour obtenir un chemin précis.

3 Efficacité de l'apprentissage

Afin de voir si ce que nous avons fait est réussi, on décide de comparer le taux de réussite d'un joueur qui joue aléatoirement et d'un joueur qui utilise notre apprentissage sur 10000 parties.

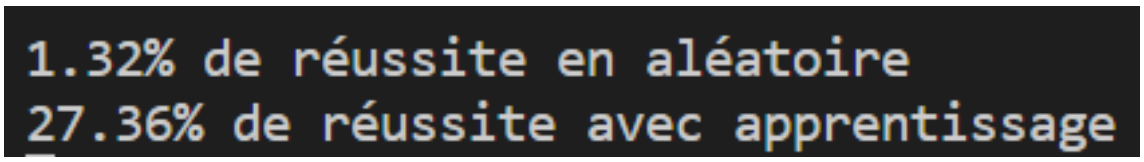


FIGURE 14 – Test 1 : Taux de réussite sur 10000 parties

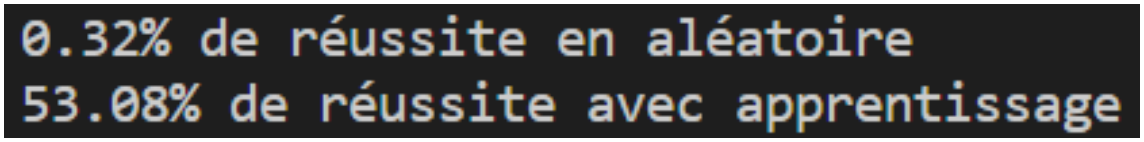


FIGURE 15 – Test 2 : Taux de réussite sur 10000 parties

Le résultat est celui qu'on attendait : avec l'apprentissage, le joueur réussi beaucoup plus à gagner la partie (cela varie entre 25% et 55% de réussite). Quant au joueur qui joue aléatoirement, il a presque aucune chance de réussir (moins de 2%). On peut donc en déduire que notre méthode d'apprentissage est réussie.
Par ailleurs, on peut commenter le fait qu'on ne dépasse pas vraiment les 50% de victoire. Cela est du à l'environnement qui reste tout de même très stochastique. En effet, homer a certes plus d'une chance sur deux d'aller au bon endroit, mais cela reste une valeur assez faible et laisse beaucoup d'erreurs.

4 Changements de paramètres

Dans cette partie, on va essayer d'effectuer quelques changements au niveau de l'interface en changeant la position des obstacles ou encore de rajouter plus de récompenses dans l'environnement (Donuts) afin d'en déduire le comportement et probablement de potentiels points à améliorer.

4.1 Obstacles

En rapprochant les obstacles de la récompense et en les positionnant de cette manière, on peut constater que la recherche d'un chemin vers la récompense devient compliquée. En effet, il y a très peu d'essai concluant où Homer parvient à atteindre les donuts car le passage étant très court et Homer pouvant se tromper, il échoue donc très souvent. De plus, les Q-valeurs aux alentours du seul passage réalisable sont très inférieures aux autres valeurs de la grille, ce qui conduit l'agent à revenir en arrière au lieu de continuer son apprentissage pour trouver le chemin optimal.

Ainsi, avec des dispositions trop compliquées, il est impossible de résoudre le problème consistant à maximiser le score.

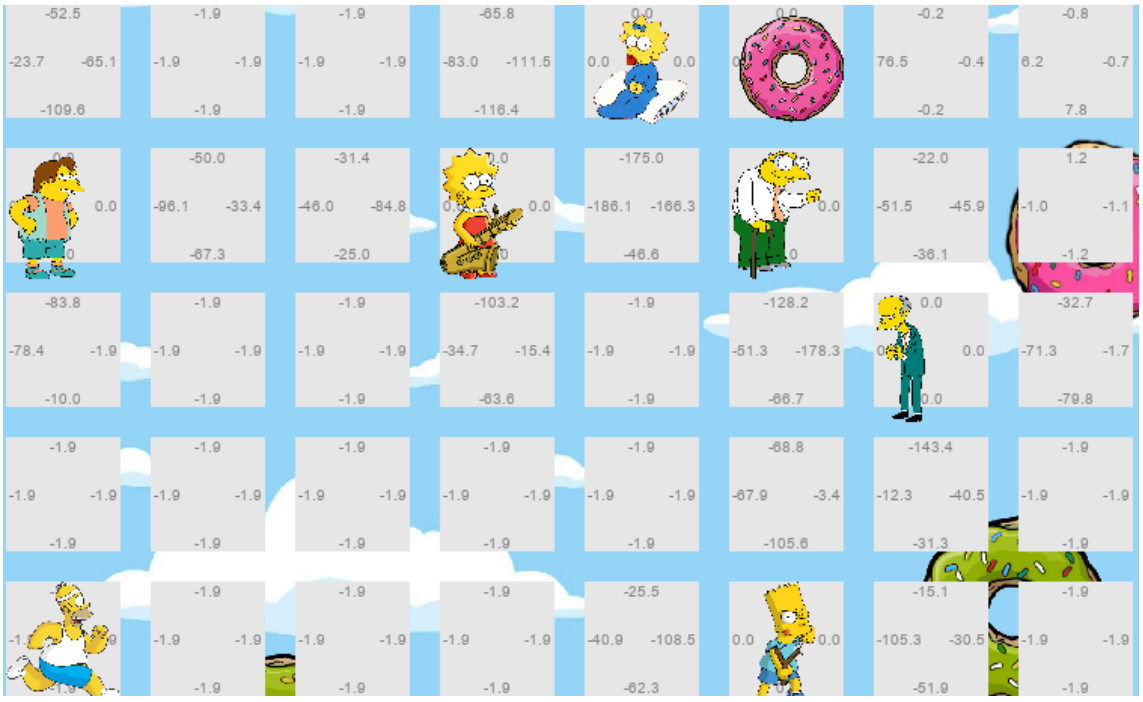


FIGURE 16 – Obstacles proches du donut

4.2 Récompenses

On reprend la même configuration que la figure précédente mais cette fois en rajoutant davantage de récompenses dans l'environnement.

L'apprentissage va donc se faire de manière à ce qu'on récupère la récompense la plus proche et présentant le moins d'obstacles autour. Par exemple dans cette figure, on peut constater que le dernier donut présent dans la première ligne n'a donc jamais été visité durant l'apprentissage.

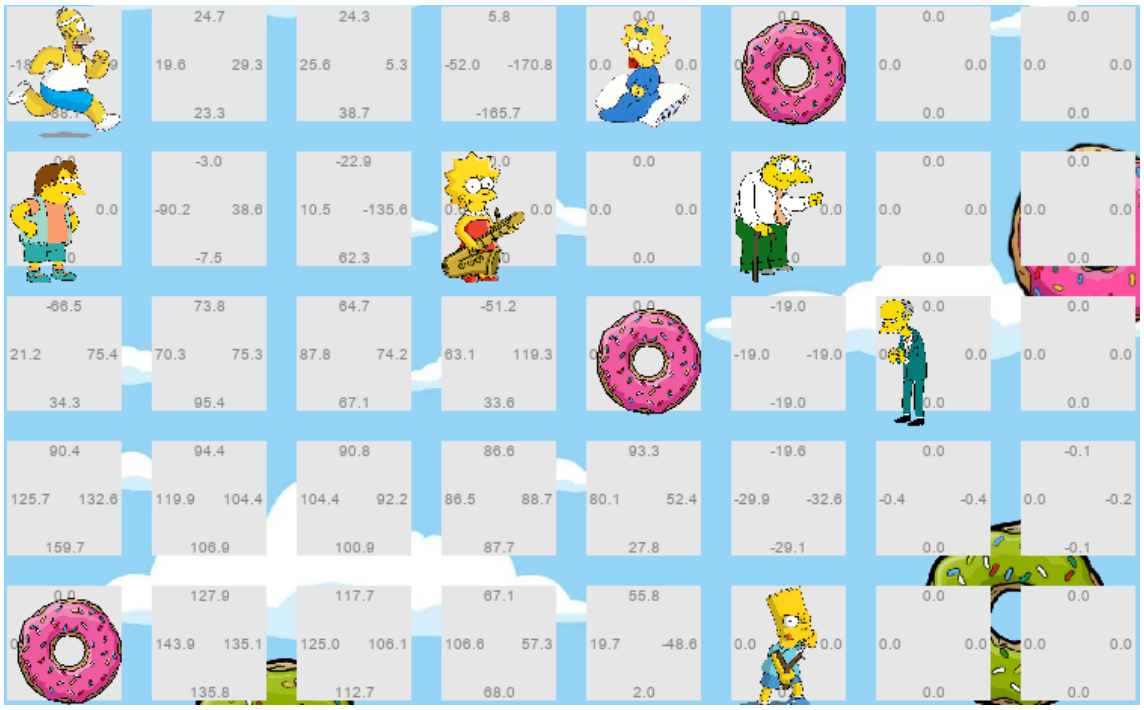


FIGURE 17 – Ajout de donuts

$\Longrightarrow$ Ces deux différents tests nous montrent que la capacité d'apprentissage reste limitée dans certains cas.

Conclusion

Conclusion du projet

Notre objectif principale était de chercher un moyen d'apprentissage pour résoudre notre problème et trouver le chemin le plus optimisé afin de maximiser le score. Nous avons utiliser les interfaces graphiques grâce à Pygame pour mieux visualiser le jeu.

Ainsi, nous avons pu mettre en place une méthode d'apprentissage automatique en utilisant le processus de Markov et l'équation de Q-learning. Il s'est avéré que l'application de cette méthode a correctement fonctionné puisque nous avons observé une convergence de notre problème vers un chemin optimal en choisissant au préalable les bon paramètres de notre équation.

Compétences acquises

Durant toutes nos scéances de travail sur ce projet, nous avons beaucoup appris et évolué. On peut retenir ceci :

- Travailler en équipe n'est pas chose aisée et cela nécessite une bonne organisation, coordination et cohésion d'équipe ! Cette expérience aura été bénéfique pour nous tous et chacun a su amener ses qualités personnelles.

- Nous avons complété notre bagage scientifique. En effet, ce projet fait intervenir de nombreuses notions de mathématiques, de machine learning, d'informatique et autres. On peut citer par exemple comme de nouvelles connaissances : le processus décisionnel markovien, des bases en machine learning, coder ces méthodes sur Python et aussi écrire en LaTex un compte rendu complet.

Impacts

Dans un premier temps nous pouvons dire que le machine learning a un impact négatif sur l'environnement. En effet, le Machine Learning est responsable de 2 à 5% des émissions mondiales de gaz à effets de serre. Il demande une puissance de calcul énorme afin de perfectionner les algorithmes et donc un considérable temps de calcul. Par conséquent, cela nécessite parfois des machines puissantes qui, celles-ci, consomment beaucoup d'électricité.

Néanmoins il y a des bénéfices à tirer du machine learning et de l'utilisation d'une chaîne de Markov. En effet, cela peut permettre de modéliser le nombre de malades dans une épidémie qui dépend du nombre de malade précédent et d'une part aléatoire. Nous pouvons la comprendre, l'analyser et trouver des solutions. C'est aussi utiliser pour résoudre des problèmes d'optimisation, on peut citer le célèbre exemple du problème du voyageur de commerce qui souhaite relier n villes en ayant un chemin optimal.

Bilans personnels

Mohamed

Ce projet m'a permis de découvrir et acquérir plusieurs compétences, essentielles pour ma scolarité. Cela passe par les mathématiques avec la notion des chaînes de Markov et le processus décisionnel de markov et dans le domaine informatique avec l'apprentissage d'un nouveau paradigme de programmation à savoir, la programmation orienté objet sur le langage python. Ce dernier sujet pourra m'être très utile dans mes projets futurs. J'ai également exploité le concept du machine Learning et ses différents modèles, qui est l'un des piliers de l'intelligence artificielle. Ce concept nous a été donc très utile lors de la création de notre jeu et son bon fonctionnement.

Maryam

Ce projet m'a appris beaucoup de choses et je l'ai particulièrement trouvé intéressant dans un premier temps, d'un point de vue compétences informatiques et universitaires. En effet j'ai pu approfondir mes compétences en python, notamment avec la programmation orientée objet que je ne maîtrisais pas autant auparavant. Elle s'avère très utile dans la Data Science pour manipuler un objet/agent lui-même et ses données. Cela m'a permis également de comprendre d'avantage les chaînes de Markov et les mettre en pratique puisque c'est un cours que nous voyons en Processus Stochastique avec Madame Bonnet. Et enfin ce projet m'a fait découvrir encore plus l'apprentissage automatique qui a été très utile dans notre société et qui l'est encore aujourd'hui. J'ai pu effectivement découvrir une de ses applications et pu la mettre en pratique. J'ai aussi appris que différents modèles existent en apprentissage automatique, tel que le model based et model free, et qu'il y avait plusieurs méthodes pour les implémenter. Dans un deuxième temps, d'un point de vue sociale, j'ai trouvé ce projet enrichissant et j'ai apprécier le travail en équipe. Je trouve que c'étais une bonne expérience car il y avait une bonne entente dans le groupe et nous avons pu chacun évoluer en mettant nos connaissances en commun.

Alexandre

Ce projet m'aura permis d'en apprendre plus sur le machine learning, ainsi que d'apprendre à utiliser ou maîtriser des librairies pythons telles que pygames ou numpy. J'ai aussi eu l'occasion de perfectionner mes connaissances concernant les chaînes de Markov, très importantes pour ce projet. Ce projet d'initiation étant principalement centré autour de l'informatique et des mathématiques, j'ai pu comprendre les concepts et les appliquer sans trop de mal.

Alexia

Au cours de ce projet, j'ai améliorer mon bagage scientifique et particulièrement en mathématiques et informatique. Nous avions commencé le projet avant le cours sur les chaînes de Markov ce qui a fait que l'on avait appris beaucoup de chose sur ce sujet. De plus, je n'avais jamais fait de machine learning et donc tout ce que nous avons fait sur l'apprentissage, comme par exemple le Q-learning, a été très enrichissant. Je maîtrise tout ce qu'il y a dans ce projet et peut-être même d'avantage grâce à des recherches complémentaires et je pense que cela me servira un jour.

Dans l'ensemble, j'ai trouvé intéressant la totalité du projet, que ça soit le principe décisionnel markovien et ses particularités, les différentes méthodes d'apprentissage, ou encore tout ce qui est en rapport à la partie implémentation des fonctions et création d'une interface

graphique pour le jeu. L'expérience acquise lors de ce projet me sera certainement précieuse pour mon avenir dans le secteur des mathématiques et de l'informatique.

Annexes

Définitions

Programmation dynamique

La programmation dynamique consiste à résoudre un problème en le décomposant en sous-problèmes, puis à résoudre les sous-problèmes, des plus petits aux plus grands en stockant les résultats intermédiaires.[4]

Apprentissage par renforcement

L'apprentissage par renforcement est une méthode utilisée pour créer des intelligences artificielles. On appelle **agent autonome** la machine (robot ou autre) à qui on fait subir cet apprentissage. Grâce à des expériences répétées un grand nombre de fois, l'agent cherche à maximiser la récompense qui lui est attribuée quand il réussit la tâche que l'on veut qu'il effectue.[2]

Environnement stochastique

Un environnement stochastique est un environnement dont les paramètres évoluent de manière aléatoire. Cela s'oppose aux environnements dits déterministes. Les organismes vivant dans ces habitats variables et imprévisibles doivent donc s'adapter rapidement afin d'y survivre et de pouvoir s'y reproduire.[3]

Modèle

Un modèle est un environnement dans lequel un agent évolue. On sépare habituellement deux manières d'effectuer un apprentissage, désignés par les termes anglais : model based et model free.

1. Model based
   Dans une simulation dite "model based", l'agent peut construire son propre modèle d'environnement. Il doit estimer lui même les probabilités de transition et les récompenses en fonction de son vécu et construit sa carte.
2. Model free
   Dans une simulation dite "model free", l'agent évolue à l'aveugle et ne connaît la récompense d'une action que quand il l'effectue. Pour trouver le chemin optimal, il doit donc effectuer un grand nombre de répétitions de l'expérience avant d'avoir des données suffisantes pour pouvoir savoir où aller. Ce genre de modèles n'est régi que par l'aléatoire, contrairement au précédent.

Lien git

L'ensemble du projet est sur ce git : `https://github.com/viitality/Homer-vs-Donuts`

Bibliographie

[1] Apprentissage amélioré à partir de python. https://linuxtut.com/fr/56f6c0b2f4cfd28dd906/, 2021.

[2] Jean benoit Delbrouck. Apprentissage par renforcement. https://jbdel.github.io/teaching/nn2019/rl1.html, 2019.

[3] Wikipedia. Adaptations aux environnements stochastiques. https://fr.wikiversity.org/wiki/Adaptations_aux_environnements_stochastiques#:~:text=Un%20environnement%20stochastique%20est%20un,de%20pouvoir%20s'y%20reproduire., 2021.

[4] Wikipedia. Programmation dynamique. https://fr.wikipedia.org/wiki/Programmation_dynamique, 2021.

[5] Wikipedia. Processus de décision markovien. https://fr.wikipedia.org/wiki/Processus_de_d%C3%A9cision_markovien, 2022.

[6] Wikipedia. Q-learning. https://fr.wikipedia.org/wiki/Q-learning, 2022.