Projet N°12 : Modélisation de comportement du rat par apprentissage par renforcement

M1 Informatique

Hector KOHLER
Aymeric MONCHI
Groupe 12

UFR INFORMATIQUE MASTER 1
SORBONNE UNIVERSITÉ

9 mai 2021

1 Introduction

Le but de ce projet est de vérifier si l'apprentissage par renforcement est une méthode adéquate pour modéliser le comportement d'un rat. Pour cela, on modélise l'expérience d'une piscine de Morris, utilisée expérimentalement pour évaluer la mémoire spatiale d'un rat. Cette modélisation repose sur des neurones dits "cellules de lieu" dont l'activité dépend de la position de l'animal dans l'espace. Dans ce projet, on utilisera l'algorithme SARSA pour modéliser l'apprentissage de l'animal comme un apprentissage par renforcement, en se basant sur une politique "ε-glouton" . L'expérience de la piscine de Morris se définit ainsi : on introduit un rat dans une bassine d'eau et ce dernier doit nager jusqu'à une plate-forme pour ne pas se noyer. Le rat est introduit à un lieu aléatoire dans la piscine (sauf au niveau de la plate-forme), et on observe qu'après plusieurs essais, le rat gagne la plate-forme de plus en plus rapidement.

La piscine est représentée comme un plan en deux dimensions (x,y) dans lequel x et y prennent des valeurs entre 0 et 1m. La plate-forme est un cercle de 0.2m de diamètre situé à mi-distance du centre et de la paroi.

2 Méthode

2.1 Modélisation du réseau de neurones

Les neurones sont ici considérés comme ayant une activité position-dépendante, et agissent en temps que cellules de lieu. C'est à dire que la position du rat détermine le degré d'activité de chaque neurone. On considère que la piscine est divisée en 400 cases égales, soit un quadrillage 20×20. Il y a un neurone au centre de chaque case qui a une activité dans un champ de 5 cm de largeur. L'espace ayant une surface de 1 m^2, on comprend dès lors que les neurones ne sont pas strictement limités à leur case, ainsi un neurone aura une activité partielle même si le rat se trouve au-delà de la case du neurone.

Dans l'expérience de la piscine de Morris, on observe sur le rat réel un raccourcissement du temps de latence, défini comme le temps que met le rat à rejoindre la plate-forme à partir de son point de départ. Cela suggère que le rat dispose d'une mémoire spatiale et d'une mémoire de travail, lui permettant de se souvenir de la position de la plate-forme par rapport au reste de l'espace, grâce aux cellules de lieu, et de se diriger en conséquence, grâce aux cellules motrices.

2.2 Algorithme d'apprentissage

Pour apprendre ces compétences, on se sert de l'algorithme SARSA, un algorithme d'apprentissage par renforcement qui va permettre d'optimiser l'activité motrice pour atteindre la plate-forme le plus rapidement possible au fur et à mesure des essais du rat. On modélise le problème comme un MDP (Markov Decision Process). Il existe plusieurs états s dans lesquels on peut effectuer des actions a. À chaque couple (s, a) est associé une

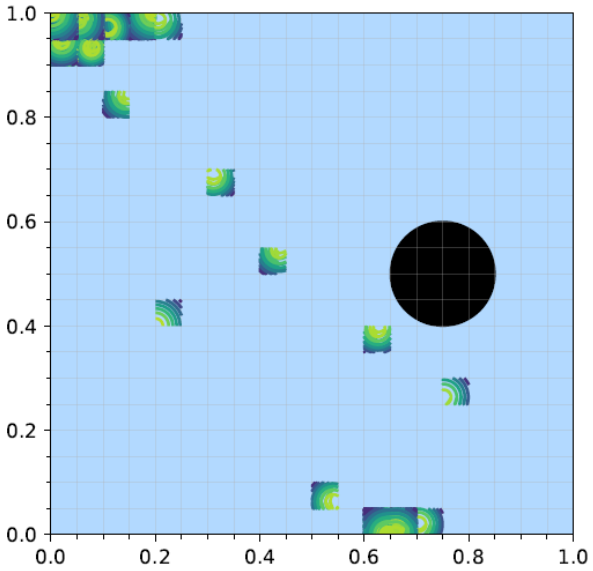


FIGURE 1 – Champs récepteurs des neurones activés par un rat, mesures prises à 50 pas d'intervalle

récompense R (le rat effectue une action, il reçoit un stimuli). SARSA vise à maximiser les récompenses obtenues en fonction des états parcourus et des actions effectuées. Pour cela il faut donner à notre rat un comportement type : une politique. On cherchera à améliorer cette politique au fur et à mesure des itérations de l'algorithme. On peut distinguer deux politiques extrêmes : l'approche "exploration" qui va consister à effectuer des actions aléatoirement dans chaque état pour connaître leur récompense, et l'approche "exploitation" qui préfère l'action de plus grande récompense dans l'état actuel. Ces deux approches présentent des inconvénients : la première ne permet pas de mettre à profit l'apprentissage, tandis que la seconde se contente d'un gain qui n'est pas forcément maximal.

La solution à ce problème est la politique dite ε-greedy, ou ε-glouton. Cette dernière est un compromis entre exploitation et exploration : on tire au hasard un réel $b \in [0; 1]$, si b dépasse le seuil $\varepsilon \in [0; 1]$ alors on choisira d'exploiter le meilleur choix trouvé dans les essais précédents. Dans le cas contraire, on explorera une autre action au hasard.

2.2.1 Application de l'algorithme SARSA à la modélisation

Dans le cas présent, on peut modéliser la politique par la fonction-valeur Q(s,a) qui, à un état s, associe une valeur pour chaque action possible. Une grande valeur d'action signifie qu'effectuer l'action a dans l'état s rapportera une grande récompense. Dans notre cas on a 400 états (nombre de neurones) et 8 actions possibles (on choisit une direction parmi 8 pour chaque déplacement). Q est donc de taille 400×8. La fonction valeur est telle que :

$$Q(s, a) = r_a^{action}(\vec{r}^{\,lieu})$$

s étant l'activité des cellules de lieu. L'apprentissage par renforcement cherchant à maximiser la récompense, on donne une récompense positive R=20 si le rat atteint la plate-forme et une récompense négative R=-5 s'il atteint un mur.

2.2.2 Choix d'epsilon

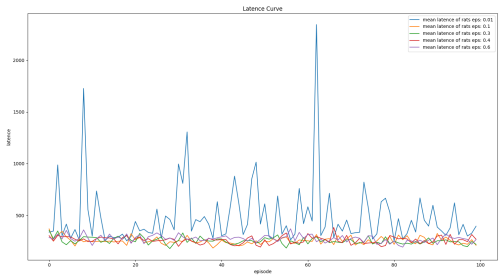


FIGURE 2 – Comparaison de la latence moyenne de 50 rats sur 30 essais selon le paramètre epsilon choisi

On constate que le temps de latence est à peu près stable si epsilon prend des valeurs entre 0.1 et 0.6, ce qui n'est pas le cas pour ε= 0.01. Par compromis, on choisit donc de fixer epsilon à 0.3. La fonction $Q(s, a)$ est mise à jour au fur et à mesure des essais grâce à l'algorithme SARSA comme suit :

$$Q(s,a) \leftarrow Q(s,a) + \Delta Q(s,a) = Q(s,a) + \eta[r + \gamma \cdot Q(s',a') - Q(s,a)]$$

Avec η le taux d'apprentissage fixé à 0.01 et γ un facteur de dévaluation fixé à 0.9 qu'on utilise pour s'assurer que Q(s,a) soit convergente. On pourra analyser des résultats sans apprentissage en fixant η= 0. En effet :

$$\eta = 0 \Rightarrow \Delta Q = 0 \Rightarrow Q(s,a) \leftarrow Q(s,a) + 0$$

Or si $\Delta Q = 0$ alors on ne met jamais à jour la fonction valeur, il n'y a donc pas d'apprentissage

2.3 Code

Pour modéliser la piscine de Morris et lancer une expérience d'apprentissage, nous avons choisi une architecture de code typique de l'étude d'algorithmes d'apprentissage par renforcement. La piscine est un environnement simulé avec lequel un agent (un rat) va pouvoir interagir. À chaque pas de temps, le rat effectue un déplacement dans l'environnement selon une action (une direction). En retour, il reçoit une récompense et fait une observation (sa nouvelle position). Et ce, jusqu'à la fin de la simulation : après un certain temps écoulé, ou quand le rat obtient une certaine récompense. On choisit d'arrêter la simulation quand un rat atteint la plate-forme ou quand il a passé plus de 10 minutes dans la piscine. Pour les visualisations, les tracés de trajectoires se font à partir du même point de l'espace $(0.5, 0.9)$ pour faciliter les comparaisons de celles-ci.

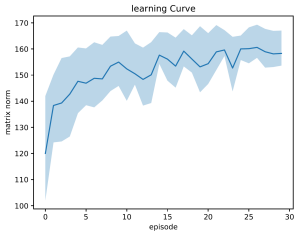
3 Résultats

3.1 Résultats sur l'ensemble des sujets

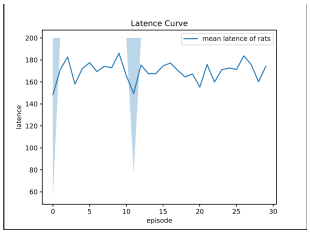
On modélise 50 rats ayant droit à 30 essais. Pour évaluer l'efficacité de l'algorithme, il faut le comparer à un témoin : on choisit donc de présenter d'abord les résultats sans apprentissage puis ceux avec. Deux critères permettront de les départager :

-D'abord la "learning curve" qui montre la mise à jour de la fonction Q(s,a) : on calcule la norme de la matrice après chaque essai. Cette fonction doit être croissante. On présentera la learning curve moyenne des rats avec une zone encadrant les valeurs du premier au troisième quartile.

-Ensuite on analysera la "latency curve", soit le temps moyen pris par un rat pour atteindre la plate-forme. On la présentera de la même manière que la précédente.

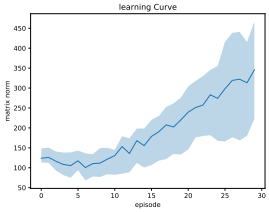


(a) courbe d'apprentissage

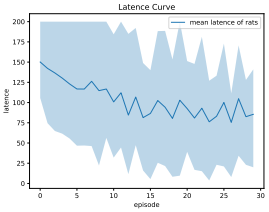


(b) courbe de latence

FIGURE 3 – Résultats obtenus sans apprentissage ($\eta = 0$)



(a) courbe d'apprentissage



(b) courbe de latence

FIGURE 4 – Résultats obtenus avec apprentissage ($\eta = 0.01$)

Sans surprise, la latence est quasi-constante et la learning curve progresse faiblement dans le groupe témoin (figure 3). On constate cependant dans le groupe apprenant une augmentation de la learning curve, ainsi qu'une diminution du temps de latence (figure 4).

En revanche, on peut noter que le modèle n'est pas aussi performant que le rat réel : en effet le rat réel peut passer de 80 à 10 secondes ($-87,5\%$) en 15 essais, tandis que le modèle passe de 150 à 100 secondes en moyenne (-33%) après 30 essais (cf page 1 projet). On peut l'expliquer notamment par une déviation standard très élevée du modèle, dont le temps de latence est de 25 secondes au premier quartile et de 150 secondes au troisième. L'apprentissage n'est donc pas optimal.

On peut expliquer la déviation standard par le choix d'un compromis par certains rats, qui chercheront à atteindre une récompense nulle, meilleure situation qu'une récompense négative. Ces rats préféreront rester dans l'eau et rester à R=0 plutôt que de prendre le risque de rentrer dans un mur et obtenir R=-5. Le rat aura ainsi une grande latence mais une récompense non négative, ce qu'il peut considérer comme un comportement adéquat.

3.2 Comparaison des cartes de navigation entre l'individu le plus performant et le moins performant

Le meilleur et le pire sujet sont déterminés à partir de la moyenne des durées de leurs essais, la plus élevée détermine le pire rat et la moins élevée le meilleur rat. Pour comparer les deux sujets, on analysera directement leur trajectoire, avant et après apprentissage :

(a) meilleur rat avant apprentissage (b) meilleur rat après apprentissage

FIGURE 5 – cartes de navigation du meilleur rat

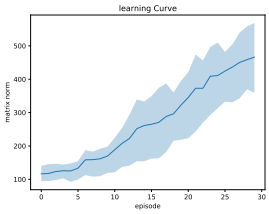
(a) pire rat avant apprentissage (b) pire rat après apprentissage

FIGURE 6 – cartes de navigation du pire rat

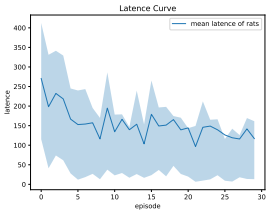
Le meilleur rat atteint la plate-forme dans les deux cas (figure 5). Mais le trajet est plus procédurier après apprentissage : le rat rejoint par lignes la plate-forme après un passage hasardeux contre le mur de la piscine. Avant apprentissage, la trajectoire est plus hasardeuse. On peut faire un constat analogue avec le pire rat : son trajet est hasardeux avant apprentissage et plus procédurier après. On en conclut que le rat a bien appris une stratégie pour rejoindre la plate-forme.

3.3 Ajout d'un obstacle sur le trajet du rat

Par la suite, on évalue les mêmes données en présence d'un objet rectangulaire sur la trajectoire du rat. Le rat ne peut pas traverser cet objet et on fixe R=0 la récompense si le rat atteint l'objet :

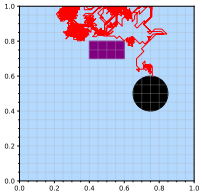


(a) Apprentissage

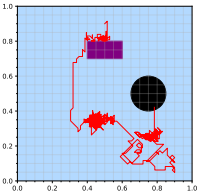


(b) Latence

FIGURE 7 – Données générales avec obstacle

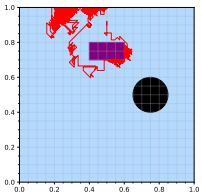


(a) Meilleur rat avant apprentissage

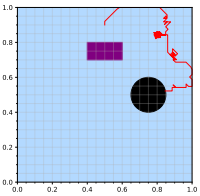


(b) meilleur rat après apprentissage

FIGURE 8 – carte de navigation du meilleur rat avec obstacle



(a) Pire rat avant apprentissage



(b) Pire rat après apprentissage

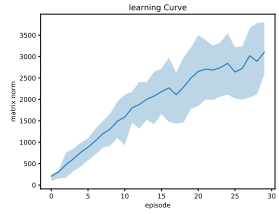
FIGURE 9 – cartes de navigation du pire rat avec obstacle

On constate que l'algorithme est robuste : on observe une diminution de la latence d'environ 100 secondes entre le premier et le dernier essai. On remarque également que le pire rat et le meilleur rat arrivent tout deux à atteindre la plate-forme après apprentissage, ce qui n'était pas le cas du pire rat avant apprentissage.

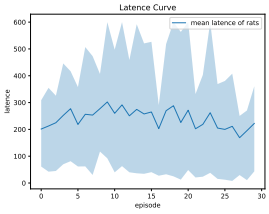
3.4 Piste d'amélioration

On peut tenter d'affiner le modèle en donnant une récompense négative au fait de rester dans l'eau : cela simule l'aversion naturelle du rat réel pour l'eau. Si la croissance de la learning curve est satisfaisante, les rats n'atteignent pas la plate-forme et la latence reste stable (voir figure 10), le modèle n'est donc pas plus affiné que précédemment. Le rat artificiel préfère atteindre le mur et obtenir R=-5 que rester dans l'eau et obtenir

$R = -1 * \Delta t$ (figures 11 et 12) . Notons que cette expérience reste réaliste dans la mesure où un rat fatigué pourrait préférer rejoindre la paroi que de nager jusqu'à la plate-forme.

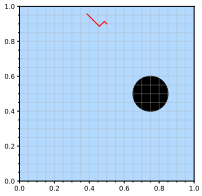


(a) apprentissage

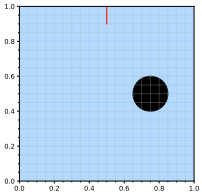


(b) latence

FIGURE 10 – données générales avec aversion pour l'eau

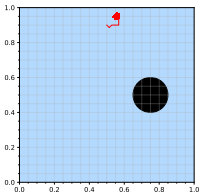


(a) Meilleur rat avant apprentissage

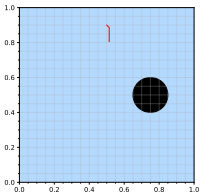


(b) Meilleur rat après apprentissage

FIGURE 11 – Cartes de navigation du meilleur rat



(a) Pire rat avant apprentissage



(b) Pire rat après apprentissage

FIGURE 12 – Cartes de navigation du pire rat

3.5 Conclusion

Si l'algorithme SARSA est efficace pour diminuer le temps de latence, il demeure, dans ce modèle, très inférieur aux performances obtenues par un rat réel. Plusieurs facteurs peuvent expliquer ce fait : d'abord, il a ici été supposé qu'il y avait indépendance entre les cellules de lieu, or on sait que les neurones réels ont des synapses qui vont stimuler les cellules avoisinantes et inhiber les cellules à distance. Ensuite, le nombre de cellules de lieu a été fixé arbitrairement, il est probable qu'un rat réel affecte davantage de cellules à la position. Ajoutons qu'on pourrait mieux optimiser les paramètres d'expérience (γ, η, ϵ). Enfin, certains paramètres ont été simplifiés : Le rat ne peut se déplacer que dans 8 directions. De plus, dans une expérience de piscine de Morris habituelle, le rat peut voir dans la pièce un certain nombre de repères (ex : un bureau) qui l'aident à s'orienter. En conclusion, si le modèle d'apprentissage par renforcement est prometteur, il est encore très éloigné des performances d'un rat réel et nécessite d'être affiné.