Apprentissage conditionné par des buts en BBRL

Cahier des charges - PANDROIDE 2023

Encadrant :
SIGAUD Olivier

Étudiants :
CELLIER Roxane
FU Zhenyue
QIN Yi

Contexte & Objectifs

L'objectif de notre projet est d'étudier différentes méthodes d'apprentissage par renforcement. En commençant par des méthodes tabulaires simples (Q-Learning par états-actions), nous devons étendre nos codes et algorithmes pour nous permettre d'implémenter et d'étudier des techniques plus poussées. Le principal algorithme autour duquel nous allons centrer nos efforts se nomme Hindsight Expérience Replay, que nous tenterons d'implémenter suivant différents modèles. Les tests de nos programmes et de nos algorithmes se feront par la recherche par un agent d'un chemin vers un (ou plusieurs) but(s) dans un labyrinthe. Nos labyrinthes seront générés aléatoirement par un module externe, et seront représentés sous forme de quadrillages. Ils pourront également contenir des murs, pour bloquer et complexifier les chemins menant à l'objectif. Nous cherchons ainsi à apprendre la meilleure politique de déplacement pour notre agent au sein de cet environnement.

Besoins

Une grande partie des algorithmes sur lesquels nous devons nous attarder ne nous ont jamais été présentés pendant notre parcours. Pour nous permettre une bonne implémentation et utilisation de ces différents algorithmes, nous devons en premier lieu les étudier attentivement. Nous avons pour ce faire lu avec attention un certain nombre d'articles qui nous étaient fournis ou que nous avons dû trouver par nous-mêmes. La plupart de ces articles ont par ailleurs été présentés dans notre carnet de bord.

Une autre étude indispensable a été faite sur de multiples librairies de Python. En effet, nous utiliserons à de multiples reprises des modules proposés par Python et d'autres développeurs externes, que nous devons comprendre pour nous assurer une utilisation correcte. Les principales librairies sont numpy pour le travail sur tableau et PyTorch pour l'utilisation de réseaux de neurones, mais aussi Gym proposée par OpenAI ainsi que BBRL développée par des chercheurs de l'université, et sur laquelle nous devrons nous appuyer pour nos futures implémentations.

Approches

Q-Learning

Le Q-learning est une méthode d'apprentissage par renforcement, qui tire son nom de la fonction Q utilisée pour le calcul de récompense d'une action exécutée dans un état donné. Cette fonction calcule donc le gain potentiel de cette action, et permet de supposer à chaque instant quelle est la meilleure action à prendre pour maximiser la récompense totale. Ce gain potentiel est stocké dans une Q-table. La phase de calcul s'effectue de manière assez simple. On considère tout d'abord une récompense de valeur 1 sur l'état but, et de 0 partout ailleurs. Lorsque ce but est atteint, la récompense est rétro-propagée sur le chemin, en fonction de deux valeurs : le facteur d'apprentissage et le facteur d'actualisation. Le premier facteur détermine à quel point la nouvelle information calculée va remplacer la précédente. Le second détermine l'importance des

récompenses à venir dans le calcul de la récompense actuelle. C'est également ce gain qui nous aide à choisir l'action à effectuer dans chaque état.

Le choix de l'action optimale peut se faire par plusieurs stratégies, et nous en considérerons deux en particulier : ε-greedy et softmax. La première permet de choisir avec une forte probabilité la meilleure action proposée par la Q-table, ou parmi les autres actions de façon équitable (mais avec une très faible probabilité). La seconde permet de choisir parmi les actions selon des probabilités proportionnelles à leurs gains potentiels. Ainsi, une action avec un gain moyen aura plus de chance d'être tirée qu'une action avec un faible gain, mais moins de chance qu'une action avec un gain plus élevé. Le choix de l'action dépendant des probabilités, il est intéressant de noter que ces méthodes convergent sur le long terme vers une politique de déplacement optimale.

Dans notre cas, nous devons considérer deux versions du Q-Learning. Dans un premier temps, nous devrons étudier une représentation en 2 dimensions, pour le cas d'un seul objectif à atteindre. Notre Q-table doit donc prendre en entrée deux valeurs, l'état courant et l'action envisagée, pour en calculer la récompense. Dans un second temps, nous devrons nous pencher sur une représentation en 3 dimensions, dans un contexte multi-objectifs. La récompense ne dépend donc plus que de l'état et de l'action, mais également du but à atteindre dans l'environnement. Nous considérons dans ce cas là chaque état atteignable comme un potentiel but. Le temps de travail pour l'algorithme avant de trouver une politique convenable étant décuplé sous cette représentation, nous travaillerons sur de plus petits labyrinthes.

HER

Hindsight Experience Replay est un algorithme ayant pour objectif d'accélérer la découverte d'une politique optimale dans un contexte multi-objectifs. Le principe de base est simple : lorsque l'on ne parvient pas à atteindre le but défini à l'origine, on considère alors l'état final comme s'il était notre objectif, et l'on recalcule alors nos récompenses en considérant ce nouvel objectif. Les récompenses ne sont donc plus calculées pendant la phase de marche, mais une fois l'épisode fini.

Pour nous permettre de calculer ces nouvelles valeurs à posteriori, il faut stocker les transitions de la marche. Nous utilisons donc pour cela un Replay Buffer, dans lequel nous stockons à chaque pas les données nous permettant le calcul (état de départ, action effectuée, état d'arrivée, récompense, et but considéré). Aux transitions stockées de cette manière, nous allons ajouter de nouvelles transitions à partir de ces dernières, auxquelles nous allons modifier le but considéré pour le remplacer par l'état atteint. Nous re-traitons ensuite ces étapes pour ainsi calculer leurs nouvelles récompenses associées. Ainsi, n'importe quelle séquence d'actions dans l'environnement permet d'améliorer la politique de déplacement. C'est cette approche que nous allons tenter d'implémenter en premier lieu.

Une version plus complète de cet algorithme consiste, en plus de considérer l'état final, à étudier également les états traversés pendant la marche. De façon aléatoire, nous sélectionnons un ensemble d'états que nous utiliserons pour remplacer sur chacune des transitions le but considéré. Ensuite, nous tirons au hasard des transitions parmi l'entièreté des valeurs stockées, et c'est sur ce sous-ensemble que nous recalculons les récompenses. Cela permet d'améliorer la politique pour les états traversés par la marche. Ainsi, une seule marche peut permettre de calculer des gains

potentiels pour plusieurs états, actions, et buts considérés. Nous devrions ainsi avoir besoin de moins de trajets pour obtenir une table des récompenses convenable. Le nombre de calculs effectués risque cependant d'être plus long et peut en effet rallonger la durée d'exécution de notre algorithme. Une fois une bonne compréhension du principe de HER, c'est cette version que nous tenterons d'implémenter.

Comparaisons

Notre objectif dans cette première partie de projet est de nous permettre de comparer entre elles les différentes approches grâce à notre implémentation. Comme annoncé en introduction, nous testerons notre algorithme dans l'optique de recherche de politique de déplacement conditionné par les buts, dans un environnement de type labyrinthe. Cet environnement sera dans un premier temps traité comme un environnement discret. C'est-à-dire que chaque état sera représenté par une case numérotée, de même que les actions discréditées selon les quatre points cardinaux.

Notre application se fera méthodiquement. Nous utiliserons plusieurs hyper-paramètres pour nous permettre d'étudier nos algorithmes. Dans un premier temps nous pourrions modifier la topologie de notre labyrinthe. Un paramètre intéressant est le ratio du nombre de murs, c'est-à-dire de cases infranchissables. Il nous est ainsi possible de créer des labyrinthes imparfaits ou possédant des entonnoirs, ou d'autres topologies spécifiques intéressantes pour l'évaluation de nos algorithmes.

Dans le gros de notre étude cependant, nous tenterons d'observer l'influence des paramètres des algorithmes en eux-mêmes. Le nombres d'épisodes correspond aux nombres de marches différentes effectuées dans le labyrinthe. Le nombre de steps correspond au nombre de pas maximum effectués au sein d'un épisode. En effet l'épisode continuerait tant que le but n'est pas atteint, pour ne pas prendre le risque d'un épisode infini, nous définissons une valeur limite. Nous pouvons également nous intéresser à la taille de l'échantillon considéré pour le recalcul des récompenses, choisis à partir des transitions du Replay Buffer.

Ouverture

Notre premier objectif est donc de comprendre et développer ces approches et algorithmes, pour nous permettre de les comparer et d'étudier leurs différences tant par leur exécution que par leurs résultats. Une fois cette première étape terminée et maîtrisée, une ouverture nous est proposée.

En nous basant sur bbrl, une librairie python existante et développée par des enseignants-chercheurs du laboratoire de l'ISIR, nous devrons implémenter les algorithmes présentés précédemment dans un environnement cette fois continu. Pour ce faire, nous n'utiliserons plus un tableau à trois entrées, mais un réseau de neurones prenant en entrée l'état actuel et le but espéré, et nous renvoyant les gains potentiels des différentes actions, nous permettant ensuite de choisir l'action adaptée. Dans un premier temps nous devrons donc prendre en main la librairie pour une assurer par la suite une correcte utilisation des algorithmes.