Apprentissage par renforcement – Projet 421

Kilian Debraux- Hélène Fermine

Modèle : Probabiliste

Nous avons choisi le modèle Probabiliste, s'agissant d'un jeu de dés, le résultat n'est pas déterminé, le but étant de faire le plus haut score possible par rapport à son adversaire.

La première variable d'état est celle du score du joueur qui va de 0 points à 800 points
Nous avons ensuite le résultat des dés lancés par le joueur. Le domaine de chaque dé est {1, 2, 3, 4, 5, 6}.

Nous avons découpé notre modèle en 4 objets :

- Un état est une combinaison de dé obtenue, à chaque état est associée une valeur qui correspond au nombre de point de la combinaison selon les règles du 421.

- Une action est une décision prise après une combinaison obtenue.
  Nous avons cet ensemble d'actions possibles pour chaque lancé :
  - Garder tous les dés
  - Relancer 1 seul dé : Il y a une action possible pour chaque dé lancé (Si on a un 6-2-4, il y a une action "Relancer le 6", une action "Relancer le 2", et une action "Relancer le 4")
  - Relancer 2 dés : Pareil que l'item précédent, on a une action pour chaque couple de dés que l'on peut relancer.
  - Relancer tous les dés.

  A chaque action est affectée une valeur : 0 si l'on garde tous les dés, -1 si l'on relance. Cela permet de privilégier l'action de garder les dés, sachant que dans les règles du 421, les adversaires doivent faire un score supérieur ou égal au nôtre avec le même nombre de coup, garder ses dés peut donc être plus avantageux que de les relancer

- Un mouvement est une transition entre 2 états. Ce n'est pas la même chose qu'une action qui est une décision. Dans le modèle probabiliste que nous avons créé, un mouvement possède une probabilité d'arriver comme passer de 4-2-2 à 4-2-1 avec 1 chance sur 6. Dans l'exemple ci-dessus, l'action serait de relancer un 2, pour cette action, on aurait 6 mouvements possibles :
  - 4-2-2 -> 4-2-1
  - 4-2-2 -> 4-2-2
  - 4-2-2 -> 4-2-3
  - 4-2-2 -> 4-2-4
  - 4-2-2 -> 4-2-5
  - 4-2-2 -> 4-2-6

Un mouvement est alors choisi au hasard selon les probabilités d'un lancer de dé. Et va modifier l'état courant du joueur

- Un agent représente le robot joueur. Il possède différents paramètres comme le taux d'apprentissage, le taux d'exploration ou le facteur d'atténuation.

Difficultés :

La première fut la gestion du modèle probabiliste avec notamment l'affectation des bonnes probabilités à chaque composition pour chaque étape.

Comme expliciter dans la présentation du modèle, chaque état peut entrainer plusieurs actions qui elle même entraine plusieurs mouvements possibles. Dans le cas du jeu 421, nous avons 55 états possibles, que l'on retrouve dans le fichier Etat421.py, pour chacun de ses états, l'action choisi et le mouvement obtenues ont des probabilités d'obtention variables

La deuxième fut la gestion des actions : les choix d'actions et de leurs récompenses étaient multiples. Pour coller au mieux au problème du 421 on a dû partir du principe que chaque état avait sa propre valeur, et on a donc dû trouver comment implanter ce résultat à chaque étape décisionnelle de celui-ci.

La troisième fut la complexité des règles, toujours en partant de la gestion du modèle, le nombre de combinaisons possible à rendu la modélisation et l'implémentation du jeu plus complexes.

Le quatrième fut la difficulté à évaluer nos résultats puisque s'agissant d'un modèle probabiliste, le joueur (ou "Agent" dans notre programme) peut être malchanceux, ceux qui faits que même si les choix qu'il fait peuvent être considéré comme bon il peut ne pas avoir de bon résultat sur la durée d'apprentissage.

Choix

Pour le jeu du 421, nous avons décidé d'implémenter un modèle de Q-Learning. L'avantage du Q-Learning est de prévoir plus loin les différentes possibilités afin d'anticiper un maximum les issues possibles, contrairement au Model Based Program. Nous trouvions ça intéressant, cela rendait notre algorithme plus performant que de s'arrêter à l'étape N+1.

Pour les actions, nous avons choisis de les créer pour chaque état différent. Selon nous, il était difficile de créer notre modèle avec moins d'action. Pourtant nous avions réfléchis à implémenter un système plus simple d'action ("Garder tous les dés", "Lancer 1 dé", "Lancer 2 dés", "Lancer tout"), mais nous trouvions qu'un tel système ne représentait pas assez la complexité du problème et pouvait faire perdre notre algorithme en précision.

En termes de méthodologie, nous avons construit notre algorithme pour résoudre un jeu à 1 dé en premier (Il avait 3 lancés pour faire le maximum). Grâce à cela, nous avons pu construire notre modèle de base, implémenter la partie probabiliste, et les récompenses. Ensuite nous avons ajouté 1 dé au jeu pour en faire un jeu à 2 dés avec des combinaisons valant plus ou moins de points. Une fois cet ajout fait, notre problème se rapprochait énormément du problème du 421, mais en moins complexe. Nous avons donc pu implémenter le principe de combinaison de dés et améliorer le système des actions pour qu'il corresponde mieux à la complexité de la situation : Lors de la première étape il n'y avait que 2 actions possibles ("Relancer", "Garder"). L'ajout d'un dé multiplie de nombre d'actions nécessaire ("Tout garder", "Relancer dé 1", "Relancer dé 2", "Tout Relancer").

Fonctionnalités non implantés - raisons

En raison du manque de temps, il était plus simple de créer un modèle avec un taux d'exploration fixe. Nous aurions aimé pouvoir générer des fonctions qui servent de taux d'exploration, afin de faire diminuer la valeur au cours du temps. Cela aurait permis au robot de plus explorer des possibilités au début de sa simulation, et ensuite de prendre des décisions en fonction de ses connaissances lorsque qu'il connaissait mieux son environnement.

Nous n'avons pas pu implémenter le calcul de la convergence de notre modèle. Par conséquent, nous ne pouvons pas l'arrêter lorsqu'il a un taux de convergence acceptable, mais nous le faisons un nombre fixe de fois. Encore une fois, nous avions ce système pour tester et nous prévoyions de passer par la convergence plus tard, mais le manque de temps et les difficultés expliquées plus haut ne nous ont pas permis cela.

Nous aurions voulu implémenter la méthode de Monte-Carlo afin de comparer avec la méthode de Q-Learning. Nous aurions généré des politiques aléatoires pour converger ensuite vers la meilleure possible.

Questions Théoriques :

- Comment modifierez-vous votre modèle proposé (que modèle, implémentation non obligatoire) pour rendre la version plus proche du jeu réel : un dès gardé ne peut pas être relancé ? Est-ce que cela augmentera ou diminuera votre espace d'état (le nombre d'états possible) ? Est-ce que cela compliquera la phase d'apprentissage ?

On gardera le modèle probabiliste.

En empêchant la relance d'un dé gardé, on diminue les actions possibles que le joueur peut effectuer au fur à mesure des lancés.

Si on veut se rapprocher du jeu réel, on doit donc partir du principe que les combinaisons de dés possibles sont un état. On avait par exemple dans notre précèdent modèle 4-2-1 était dans le même état que 1-2-4, 2-1-4, 1-4-2, 2-4-1 et 4-1-2, ce qui nous donnait 55 états en.

tout. Pour avoir la notion de dé gardé non relancé on doit partir du principe que ces combinaisons sont dans différents états et donc avoir 216 états possibles.

En partant de 216 états possibles on a donc également une multiplication du nombre d'actions possibles, même si celle-ci seront dégressives à chaque lancer.

En multipliant les états par 4, on complexifie la phase d'apprentissage puissent que cela nécessitera plus de ressources et plus de temps pour lui permettre d'explorer tous les états possibles, il sera alors nécessaire de lancer de plus longue phase d'exploration pour avoir des résultats probants.

Contribution Personnelle :

Kilian :

J'ai bien aimé le projet qui m'a permis de vraiment comprendre les mécanismes de l'apprentissage par renforcement. C'est grâce à la pratique que j'ai pu mieux interpréter les cours magistraux. Je trouve justement dommage de ne pas avoir eu plus de tps pour nous faire pratiquer et assimiler les cours (Je pense que cela est dû en grande partie au manque de temps, au final nous n'avons eu que 24h).

J'ai trouvé très difficile la partie probabiliste du problème, nous l'avons expliqué dans ce rapport, mais cela rendait complexe l'analyse des résultats de notre algorithme, et multipliait les différentes possibilités à prendre en compte. Je pense qu'il me faudrait encore un peu de temps pour comprendre comment prendre en compte l'incertitude et les probabilités dans un modèle de RL.

Hélène :

Ce projet m'a aidé à comprendre plus facilement les théories abordées lors de ce cours. N'ayant pas un grand bagage technique avoir une réelle mise en situation comme ce projet est formateur.

J'ai beaucoup aimé l'approche probabiliste que le problème apportait, jouant régulièrement à ce jeu, je trouvais intéressant de voir si le robot allait agir comme les "vrais" joueurs agissent ou si son système de pensé était totalement différent.

J'aurais aimé pouvoir aller plus loin dans le jeu pour avoir cette notion d'adversaire multiples et de dé gardé non re lançable mais le temps impartit pour ce cours et la complexité de cette implémentation ne l'a pas permis.

L'étude plus avancé le méthode Monte-Carlo pendant le dernier cours aurait été aussi quelque chose qui aurait été intéressant de mettre en place pour mieux comprendre l'implication des valeurs d'états estimé et ainsi peut être amené à un modèle déterministe.