Apprentissage par renforcement

L'exemple du jeu de Nim en Python

Niveau : lycée
Notions : programmation en Python, calcul de probabilité, stratégie gagnante dans un jeu.

1. Introduction

Les deux programmes Python présentés illustrent l'apprentissage par renforcement en intelligence artificielle sur le jeu de Nim ou jeu des allumettes.

La règle du jeu est simple : on part avec 8 allumettes, chaque joueur·se, tour à tour, enlève une ou deux allumettes, celui·celle qui enlève la dernière a gagné. La stratégie gagnante est simple : la première personne laisse 6 allumettes à la deuxième. Quoique joue la deuxième, la première aura la possibilité de ne laisser que 3 allumettes à la deuxième. Et quoique fasse alors la deuxième, la première pourra enlever la dernière. C'est ainsi celui·celle qui commence qui est certain·e de gagner s'iel joue bien.

Le principe de l'apprentissage par renforcement est le suivant : le programme joue au hasard (soit contre l'humain·e soit contre lui-même). Si l'issue est heureuse, c'est-à-dire se termine par une victoire du programme, on augmente les probabilités de jouer la série de coups joués (récompense). Si l'issue est malheureuse, c'est-à-dire se termine par une défaite du programme, on diminue les probabilités de jouer la série de coups joués (punition). Petit-à-petit, les probabilités de jouer les bons coups augmentent et celles de jouer les mauvais coups diminuent. Le programme apprend à jouer de mieux en mieux.

C'est un type d'apprentissage beaucoup utilisé dans la prise de décision lorsque l'humain·e est incapable de la donner en la programmant directement. C'est sur la base de cet apprentissage qu'un programme a réussi à battre le champion du monde de Go en 2017.

Les programmes et activités ci-dessous peuvent être utilisés en classe seuls. Mais on peut aussi s'inspirer d'un atelier proposé à la MMI (en 2021-2022 et 2022-2023) : « Jeu de Nim et IA ». Cet atelier d'informatique débranchée permet de comprendre l'apprentissage par renforcement sans besoin de programmer. Le déroulé se trouve ici : https://mmi-lyon.fr/?site_ressource_peda=jeu-de-nim-et-ia

On peut aussi imaginer tout ce qui suit comme une excellente suite à une venue à la MMI sur cet atelier. Les programmes ont été écrits pour illustrer l'atelier ci-dessus, il est donc conseillé de l'avoir fait ou d'en avoir regardé le déroulé avant. On peut aussi aller regarder la vidéo (à venir sur le site de la MMI) pour voir comment consruire sa machine apprenant à jouer au jeu de Nim sans ordinateur.

Enfin, si vous voulez tester différents apprentissages, une application a été développée par Eric Duchêne, on peut la trouver ici : https://projet.liris.cnrs.fr/~mam/machine/

Celle-ci est commentée dans la dernière section.

2. Description des programmes Python

Les deux programmes sont largement commentés. Ils sont écrits avec la règle de jeu ci-dessus. Dans le premier, le programme apprend en jouant contre un·e humain·e. Dans le second, il apprend en jouant contre lui-même (il joue à la fois en premier et en deuxième), il n'y a donc plus d'intervention humaine.

2.1 jeudeNim_humain.py

C'est le programme dans lequel l'ordinateur apprend contre un·e humain·e. Le programme commence toujours car la première personne ayant une stratégie gagnante, si elle jouait parfaitement bien, le programme ne gagnerait jamais et ne serait jamais récompensé, il n'apprendrait rien.

Après chaque partie, le programme est récompensé s'il a gagné. On ajoute une bille de la couleur jouée dans les verres correspondants. Il est par contre puni s'il perd, on enlève alors une bille de la couleur jouée dans les verres correspondants.

La probabilité de gain donnée après chaque partie est calculée pour un·e adversaire jouant optimalement : si le programme ne laisse pas 3 ou 6 allumettes, il a perdu. Et quand il laisse 6 allumettes, on joue le coup qui laisse le plus de chances qu'il ne laisse pas 3 allumettes (en regardant le contenu des verres). Quand il laisse 3 allumettes, on lui en laisse 2 pour se laisser une chance. Cette probabilité de gain est toute théorique mais permet de voir comment la machine évolue.

Contre un·e humain·e qui ne fait pas de cadeau, la machine jouera parfaitement bien au bout d'à peu près une quinzaine de parties, voire avant. Si l'humain·e ne joue pas parfaitement bien (ou est trop gentil·le avec la machine), le programme apprendra moins vite.

2.2 jeudeNim_machine.py

C'est le programme dans lequel la machine joue contre elle-même. Dans ce cadre, elle est récompensée pour le côté qui a gagné et punie pour le côté qui a perdu. L'avantage est qu'elle apprend toute seule, sans intervention humaine. De plus, elle apprend relativement vite puisqu'elle est, après chaque partie, récompensée et punie.

La probabilité de gain donnée après chaque partie correspond à la probabilité théorique de gagner pour la machine jouant le·la premier·ère joueur·se. Encore une fois, cela permet de voir l'évolution du programme.

Le programme devient très bon au bout d'une quinzaine de parties, excellent au bout d'une trentaine, et ce sans intervention humaine.

3. Idées d'activités autour du jeu et des programmes

3.1 Découverte de la stratégie gagnante

Faire jouer les élèves pour leur faire découvrir la stratégie gagnante. Travailler la notion de stratégie gagnante : c'est une stratégie qui permet de gagner à tous les coups, quoique fasse l'autre. C'est donc un plan de jeu qui permet de gagner quels que soient les coups de l'adversaire.

On peut aussi varier les règles du jeu. Il est par exemple intéressant de commencer avec 9 allumettes (pour se rendre compte que c'est alors le·la deuxième joueur·se qui a une stratégie gagnante). On peut aussi modifier le nombre d'allumettes qu'on peut prendre : 1, 2 ou 3 (petite généralisation) ou 1, 3, 4 mais pas 2 (plus compliqué). On conseille 1, 3, 4 avec 10 verres par exemple.

3.2 Etude des programmes

Au choix : étudier les programmes joints, en écrire un qui gagne à tous les coups (sans apprentissage, en programmant la stratégie gagnante trouvée avant), modifier les programmes avec des nouvelles règles, etc.

3.3 Calcul de probabilités

Il peut être intéressant de faire calculer la probabilité que la machine gagne à la première partie (quand elle a une chance sur deux d'enlever 1 allumette dans toutes les situations de jeu) contre un·e humain·e qui joue parfaitement bien. C'est 1/8.

On peut aussi, plus difficile, faire calculer la probabilité de gagner de la machine en tant que premier·ère joueur·se lorsqu'elle joue contre elle-même (à la première partie). On peut commencer avec moins d'allumettes (par exemple 5), ce qui rend le calcul plus simple (et l'arbre de jeu moins long). On détaille ce calcul ici :

On part avec 5 allumettes. Au premier coup, ½ de jouer une allumette, ½ de jouer deux allumettes. Au deuxième coup, même chose. Ainsi, après un coup de chaque joueur·se (joué·es par le programme), on a ½ de se retrouver avec deux allumettes, ¼ de se retrouver avec 3 allumettes et ¼ de se retrouver avec 1 allumette. Quelles sont maintenant les probabilités de gagner pour le·la premier·ère joueur·se dans ces situations ?

S'il reste une allumette, gain à coup sûr.

S'il reste deux allumettes, ½ de gagner (en retirant deux allumettes, sinon, c'est perdu).

S'il reste trois allumettes, ½ de perdre directement en enlevant deux allumettes, ½ de laisser deux allumettes à l'adversaire et alors encore ½ que l'adversaire n'enlève pas les deux dernières. Au total, ¼ de gagner dans cette situation.

Au total, cela fait donc ¼ + ½ * ½ + ¼ * ¼ = 9/16.

On peut bien sûr imaginer plein d'autres activités. N'hésitez-pas, si vous en construisez, à nous les envoyer pour les partager sur cette page de ressources. Merci !

4. L'application d'Eric Duchêne

Celle-ci se trouve ici :

Voici les paramètres modifiables. Ceux-ci correspondent à tous les paramètres de l'apprentissage

par renforcement. On peut même imaginer encore plus de liberté, le·la programmeur·se en ayant plein, la qualité de l'apprentissage dépendant souvent de ces choix.

On peut :

- changer les règles en autorisant un certain nombre d'allumettes à prendre.
- changer le nombre de verres de départ.
- changer le nombre de billes dans chaque verre au départ.
- choisir le taux de renforcement et de punition.
- faire jouer la machine contre un·e expert·e (qui ne fera aucun cadeau), contre une personne jouant au hasard, contre elle-même.
- choisir si la machine commence ou pas.

Bien penser à « préparer la machine » à chaque fois qu'on change les paramètres.

Les victoires sont celles de la machine. Lorsqu'elle joue contre elle-même, les victoires sont comptabilisées pour la machine jouant en premier si vous avez coché « la machine commence ».

On peut remarquer que la machine n'apprend rien si l'expert·e a une stratégie gagnante. Exemple : 8 verres, 1 ou 2 allumettes à enlever, la machine ne commence pas.

On peut remarquer que contre de l'aléatoire, la machine n'apprend pas grand chose.

On peut aussi utiliser la machine contre elle-même pour gagner l'intuition des stratégies gagnantes dans des situations peu évidentes : par exemple avec la règle « on retire 1, 3 ou 4 allumettes ». Ensuite, on a une conjecture, une idée de la stratégie gagnante et il faut montrer qu'elle est en effet gagnante.

On peut aussi essayer de reprogrammer en Python toutes ces situations.