Clotilde Arléry *03 décembre 2023*

Rapport semaine de programmation

Introduction

	C	D
C	+2 / +2	-1 / +3
D	+3 / -1	0 / 0

Figure 1. Matrice des gains des joueurs

Nous jouons à un jeu inspiré du **dilemme du prisonnier répété** qui, en théorie des jeux, modélise une situation où deux joueurs doivent choisir de coopérer ou de trahir, les joueurs ne connaissant pas à l'avance l'action de l'autre joueur. La situation théorique optimale est une coopération mutuelle. Dans notre cas, les joueurs décident simultanément sans se concerter. Si chacun coopère, chacun perçoit deux points. Si chacun trahi, personne ne perçoit ni ne perd de point. Si l'un des joueurs trahi alors que l'autre coopère, il perçoit trois et l'autre perd un point. Les gains respectifs sont représentés sur la **Figure 1**. Une partie se joue en 100 manches.

Chaque élève de la classe développe un algorithme mettant en œuvre une stratégie dans l'optique d'un tournoi qui confrontera chacun des algorithmes. Le but étant d'avoir le meilleur score total, c'est-à-dire de maximiser son score à l'issue des 100 manches et d'avoir la meilleure stratégie pour vaincre toutes les autres stratégies. ***Un bruit*** *(une simulation d'erreur de communication par exemple : un joueur coopère mais l'action "faire défaut" est transmise par erreur) de* ***15%*** *est imposé dans l'exercice.*

1 - Théorie des jeux et dilemme du prisonnier

Dans The Evolution of Cooperation, R. Axelrod s'intéresse à ce type de dilemme du prisonnier répété (joué en plusieurs manches). Se démarque alors que les stratégies altruistes (où la coopération est de mise par défaut) sont plus performantes sur le long terme que les stratégies où chacun fait défaut. Plus tard, les travaux d'**A. Rapoport** mettent au jour une stratégie semblant vaincre toutes les autres : la stratégie du ***Tit for Tat*** (en français *oeil pour oeil*). Une variante encore plus efficace de cette stratégie émerge ensuite : il s'agit de "**pardonner**" un certain nombre de fois en cas de défection de l'autre joueur. Cette variante possède a priori deux avantages : débloquer les cycles négatifs (si l'autre joueur fait défaut une fois mais qu'il recommence à coopérer ensuite), ne pas tomber dans un cycle négatif en cas de bruit (par exemple, une défection faite par erreur ou une communication mal transmise). Nous appellerons respectivement *copycat* et *copykitten* la stratégie *Tit for Tat* et sa variante avec pardon.

2 - Notre stratégie

Pour développer notre stratégie, il nous est apparu les postulats suivants :

- Bien qu'il soit théoriquement optimal de coopérer, rien n'empêcherait un joueur de détourner la situation à son avantage : s'il est tout à fait certain que tout le monde coopère, alors un joueur particulier aurait intérêt à faire défaut à tous les coups. Le score final serait de 300 pour ce joueur particulier et de -100 pour les autres joueurs
- Bien que *copycat* et *copykitten* soient efficaces, il n'est pas garanti que ces stratégies soient adoptées par les autres joueurs.
- Il existe a priori environ autant de stratégies possibles que de joueurs.

Par conséquent, il est raisonnable de penser qu'on ne peut pas prédire la stratégie d'autrui. Plutôt que de déterminer une stratégie fixe qui n'aurait probablement pas pu englober toutes les stratégies possibles (or nous voulons gagner peu importe la stratégie adverse), nous nous sommes intéressés aux algorithmes d'apprentissage automatique qui, par définition, permettent une plus grande adaptabilité aux situations.

Nous avons pour cela exploré deux pistes. D'une part, le **Q-learning** qui est une méthode d'apprentissage automatique (*machine learning* en anglais) par renforcement très performante pour ce type de problème de classification où un gain est associé à une action. Ceci étant, l'un des points forts du Q-learning est qu'il permet de choisir l'action (qui maximise le gain) la plus probable sans connaissance initiale de l'environnement. Or, dans notre cas, c'est un énorme inconvénient ! L'historique des joueurs étant une donnée clé pour prédire l'action suivante et donc optimiser la décision et pour, de plus, déceler des schémas dans le jeu adverse.

Nous nous sommes donc tournés vers une méthode d'apprentissage profond (*deep learning* en anglais) appelée RNN (*Recursive Neural Network* en français). Un réseau de neurones récurrent est un type spécial de réseau de neurones qui a des connexions récurrentes, ce qui signifie qu'il peut conserver des informations sur des états antérieurs et utiliser ces informations pour influencer les calculs futurs, grâce à l'introduction de boucles de rétroaction dans l'architecture. Nous décrivons le fonctionnement global de notre modèle dans la Figure 2.

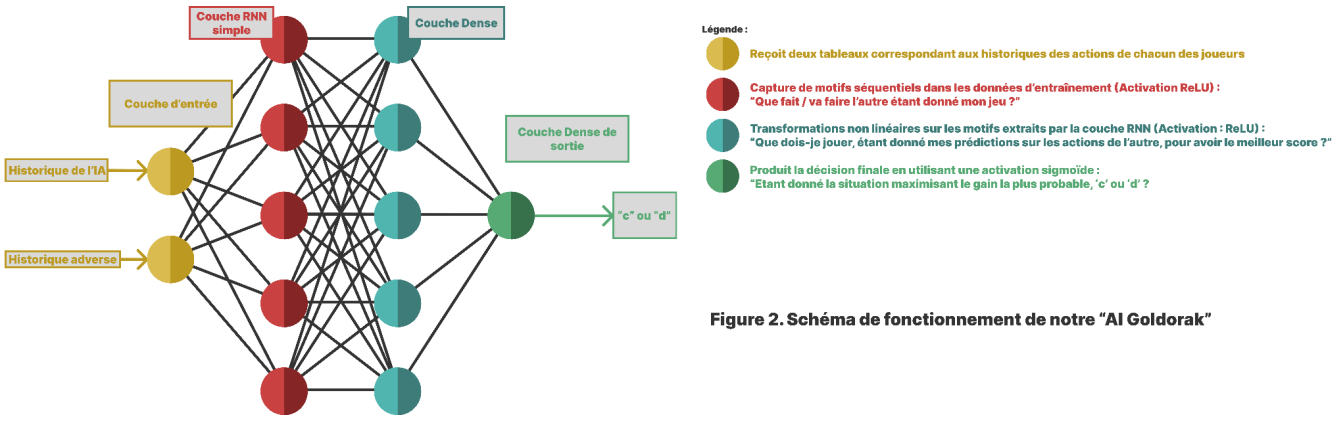

Figure 2. Schéma de fonctionnement de notre "AI Goldorak"

En particulier, pour ajuster ce modèle à notre situation, nous avons couplé une couche RNN avec une couche dite dense qui permet de capturer des relations plus complexes entre les caractéristiques apprises par la couche RNN. Ici, la couche dense permet notamment de relier les motifs d'action appris par la couche RNN et les gains que l'on peut obtenir en tirant profit de la stratégie adverse. Ces relations prenant en compte le gain sont ensuite envoyées dans la couche dense de sortie qui est en fait un unique neurone modélisée par une fonction d'activation sigmoïde (Figure 3.). Cette fonction permet de déterminer de manière statistique s'il est plus judicieux de coopérer ou de faire défaut. En réalité, dans la version finale, notre IA comporte 16 neurones RNN, 16 neurones en couche dense et 1 neurone en couche dense de sortie.

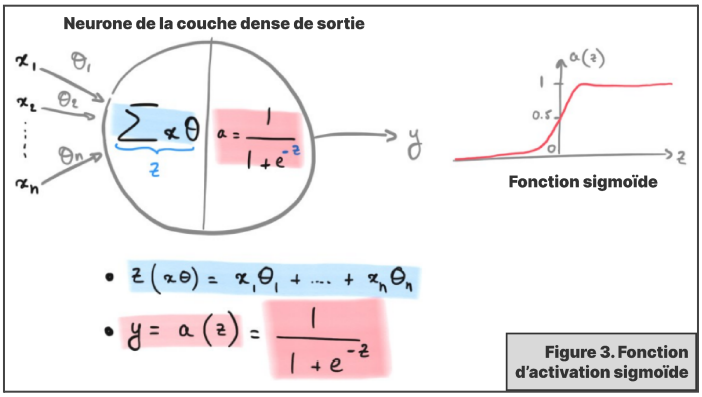

Figure 3. Fonction d'activation sigmoïde

Pour revenir sur le fonctionnement concret et mathématique du modèle, chaque neurone est représenté par une fonction d'activation permettant de déterminer mathématiquement des relations entre les entrées. En particulier, les neurones des couches RNN et de la couche dense sont modélisés par des fonctions d'activation ReLU (*Rectified Linear Unit* qui permettent d'apprendre des relations non-linéaires et donc complexes - ce qui intéressant dans notre cas). Le neurone de la couche dense de sortie est modélisé par une fonction d'activation sigmoïde (Figure 3.).

Enfin, **pour inclure les 15% de bruit**, nous avons simplement considéré qu'il y aurait du bruit 15 fois par partie. On regarde donc la longueur du tableau d'historique. Si la longueur du tableau est un multiple de 15 alors on fait l'action contraire de celle prévue. Sinon, le programme suit son cours de manière inchangée. Comme nous n'avons pas de stratégie directement identifiable par autrui et que l'autre joueur ne sait pas quand il y aura du bruit, **il ne nous a pas semblé nécessaire de randomiser l'apparition du bruit**.

3 - Performances de l'IA

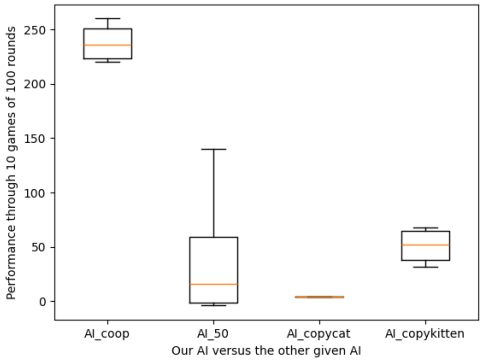

Figure 5. Performances

Si le modèle peut paraître complexe, il n'en reste pas moins **plus performant que les autres IAs** sur lesquelles nous l'avons testé, incluant les IAs développées dans le cadre du TD (AI_50, AI_coop) ainsi que deux supplémentaires (*copycat* et *copykitten*). **Et ce, malgré l'ajout d'un bruit de 15%**. Par ailleurs, pour l'améliorer, nous avons ajouté un traitement postérieur au travail de notre IA mais qui fait partie intégrante du programme, qui rectifie les réponses de l'IA si la stratégie adverse est de faire défaut constamment. En effet, pour améliorer la rapidité de traitement et ainsi s'adapter aux exigences du tournoi, nous avons dû diminuer le nombre de neurones initialement prévu, ainsi que le nombre de lots de données d'entraînement, ce qui diminue les performances de l'IA dans le cas précis où l'adversaire fait constamment défaut. Ce traitement additif est effectif si le taux de trahisons dépasse 35%. Pour évaluer les performances, nous avons confronté notre IA aux autres (cf. plus haut) lors de 5 matchs en 100 manches, et calculé pour chaque match la différence entre le score de notre IA et le score des autres. Comme l'illustre la figure 5, **notre IA est plus performante que les autres** en moyenne et dans l'absolu (car pas de valeur négative).

4 - Applications

La Théorie des jeux et en particulier le dilemme du prisonnier, trouvent des **applications très concrètes** dans la vie de tous les jours ou dans des domaines qui impactent ce quotidien. Un exemple souvent décrit en économie est la question de la spécialisation et de la concurrence, comme décrit dans la figure 4.

De même, la théorie des jeux peut être intéressante pour modéliser la gestion des ressources naturelles : la surpêche ou la surexploitation des ressources naturelles peuvent être modélisées comme des jeux. Les acteurs doivent décider s'ils doivent coopérer en maintenant des niveaux durables ou agir de manière égoïste pour maximiser leurs propres gains à court terme. Ou encore en sociologie où les conflits sociaux peuvent être analysés comme des jeux où différents groupes ou individus doivent choisir entre la coopération et la confrontation. Les manifestations, les grèves et d'autres formes de protestation peuvent être modélisées de cette manière. Enfin, en relations internationales, où les négociations et les accords entre nations peuvent être modélisés comme des jeux. Chaque pays doit décider s'il respectera ou non les termes d'un accord, en fonction de ses propres intérêts.

Figure 4. Théorie des jeux : concurrence et spécialisation

		Bala : Riz	Bala : Manioc
Anil	Riz	Chacun produit du riz : il y a surabondance de riz (prix bas) Il y a une pénurie de manioc Anil ne produit pas de manioc, la culture qu'il est le plus apte à produire	Pas de saturation du marché Prix élevé pour les deux cultures Les deux fermiers produisent la culture pour laquelle ils sont les moins aptes
Anil	Manioc	Pas de saturation du marché Prix élevé pour les deux cultures Les deux fermiers produisent la culture pour laquelle ils sont les plus aptes	Chacun produit du manioc : il y a surabondance de manioc (prix bas) Il y a une pénurie de riz Bala ne produit pas de riz, la culture qu'il est le plus apte à produire

		Bala : Riz	Bala : Manioc
Anil	Riz	Anil obtient 1 Bala obtient 3	Tous les deux obtiennent 2
Anil	Manioc	Tous les deux obtiennent 4	Anil obtient 3 Bala obtient 1