

LEPETIT Lenny

AZZOUG Dalia

MATT Lucie

NISHARIZE Jeancy Candela

Objectif du projet : Création d'un agent sous la méthode Double-Q Learning qui va apprendre à obtenir les meilleurs rendements possibles pour le dilemme du prisonnier face à des adversaires multiples et différents.

1- Introduction

Deux suspects sont arrêtés pour une infraction commise (port d'armes illégale) et les enquêteurs souhaitent leur faire avouer le délit qu'il allait commettre (braquer une banque). Si l'un des deux suspects admet aux enquêteurs que leur coéquipier était sur le point de mener une attaque, alors le suspect ayant trahi est libéré, et la personne trahie obtient une peine de 5 années de prison. Si les deux personnes se trahissent mutuellement, alors chaque suspect prend une peine de quatre années de prison. Si aucun des deux ne trahit son partenaire et coopère, alors chaque prisonnier est emprisonné pour deux années de prison. Chaque année de liberté remportée par rapport à la pire situation rapporte 1 de gain, la pire situation étant celle où l'on ne trahit pas son coéquipier et lui nous trahit. Cette situation nous donne la matrice de gain suivante :

	Coopération	Trahison
Coopération	(3 ; 3)	(5 ; 0)
Trahison	(0 ; 5)	(1 ; 1)

Quel que soit le choix défini par notre coéquipier, notre gain est plus important en le trahissant. En effet, si notre partenaire coopère, notre gain est de 3 en coopérant et de 5 en le trahissant, donc on choisit de le trahir. Si notre partenaire décide de nous trahir, alors notre gain de coopération est de 0 tandis qu'il est de 1 en le trahissant, donc on choisit également de le trahir. Cela nous amène donc à une situation sous-optimale (1 ; 1) tandis

qu'il aurait été préférable de coopérer pour obtenir un gain de (3 ; 3). En effet, chaque agent aurait eu un gain supérieur et ceci sans compromettre le bien-être d'au moins un autre agent. On appelle cette situation le dilemme du prisonnier.

Il est à noter que dans le contexte initial, cette situation est censée se présenter qu'une seule et unique fois, mais dans ce contexte nous allons essayer de l'adapter à une situation où l'on aurait affaire à un enchaînement de séquences du même type, avec des partenaires différents.

L'objectif de notre projet est de programmer un agent qui va apprendre à obtenir les meilleurs gains possible en fonction de ces adversaires. En effet, il peut exister différents types de profils en fonction de cette situation. Il se peut que notre partenaire nous fasse une confiance aveugle et décide toujours de coopérer avec nous et prend principalement dans son arbitrage le bien-être de l'équipe, mais il se peut également que notre partenaire pense avant tout à son bien-être individuel et nous trahisse. Avoir des informations concernant le profil de risque de notre partenaire peut s'avérer être une information cruciale à ne pas négliger. Pour mettre en avant cette situation, nous avons imaginé différents types de profils types :

- 1) **Le coopérant** : Cette personne coopéra toujours avec nous, quelque soit notre choix, et ce même après plusieurs trahisons.
- 2) **Le Trahisseur** : Ce partenaire nous trahira à chaque fois.
- 3) **Le Tit-For-Tat** : Au premier tour, ce coéquipier coopéra avec nous. Pour les tours suivants, il fera au Nème tour l'action au que nous avons effectué au tour N-1.
- 4) **Le Random** : Aucune stratégie ici, il y a une chance sur 2 pour qu'il nous trahisse.
- 5) **La Copie** : Cet agent sera entraîné exactement de la même manière que notre agent principal à qui l'on souhaite maximiser ces résultats. Nous allons entraîner cet agent sur les 4 premiers profils pour établir des moyennes de comportements. Ensuite nous nous serviront de cet agent pour entraîner notre agent principal.

L'objectif est d'avoir des profils variés. Il est également intéressant de voir comment notre agent principal va réagir suite à l'introduction d'un nouvel adversaire contre qui il n'a aucune information de base. Cet agent copie aura une mémoire afin d'améliorer la vraisemblance des choix de notre agent principal car l'objectif est de trouver une solution pour que notre agent s'adapte au mieux.

Par principe, nous supposons que notre agent n'est pas capable de connaître les actions de son adversaires mais à force d'entraînement il devra être en mesure d'estimer des

probabilités sur celles-ci, au moins sur les adversaires les plus simples (les quatre premiers).

Hypothèses :

Dans la logique des choses, contre les adversaires Coopérant, Trahisseur et Random, notre agent devra toujours trahir son partenaire car le gain sera toujours plus élevé dans l'immédiat et l'adversaire ne s'adapte pas. Contre l'adversaire Tit-For-Tat, nous pensons que notre agent aura des choix variés au départ le temps de comprendre que l'adversaire copie nos anciens choix, puis il décidera de toujours coopérer.

Contre l'agent Copie, il est difficile d'émettre une hypothèse. En effet, déjà il faut se demander comment mettre en avant cette adversité. Est-ce qu'on fait plusieurs rencontres entre notre agent et notre copie (par exemple une trentaine) et ensuite enregistrer ces informations de ces parties dans la mémoire de notre agent principal pour qu'il puisse modifier sa politique interne mais l'agent copie lui n'enregistre pas ces nouvelles informations. Ce qui fait que lors de la deuxième série de rencontre entre l'agent principal et l'agent copie, l'agent principal devrait tirer un avantage de ce surplus d'informations. A long terme, notre agent devrait avoir des gains plus élevés que l'agent copie.

Démarche :

Etape 1 : On fixe les règles du jeu dans notre code, on implémente les adversaires possédant des schémas heuristiques (les quatre premiers agents) et on essaye de faire en sorte que notre agent puisse s'adapter à ce type d'adversaire.

Etape 2 : On enregistre notre agent principal avec ses principales règles de décisions qu'il a apprises. On fait une copie de cet agent et on l'appelle l'agent Copie. A partir de ce moment, l'agent copie n'apprend plus mais il s'occupe que de ces politiques apprises.

Etape 3 : On définit les séquences de jeu entre notre agent et l'agent copie. A chaque fois qu'une séquence de parties est terminée, notre agent principal met à jour sa politique. Ensuite il refait une autre séquence de partie contre l'adversaire Copie.

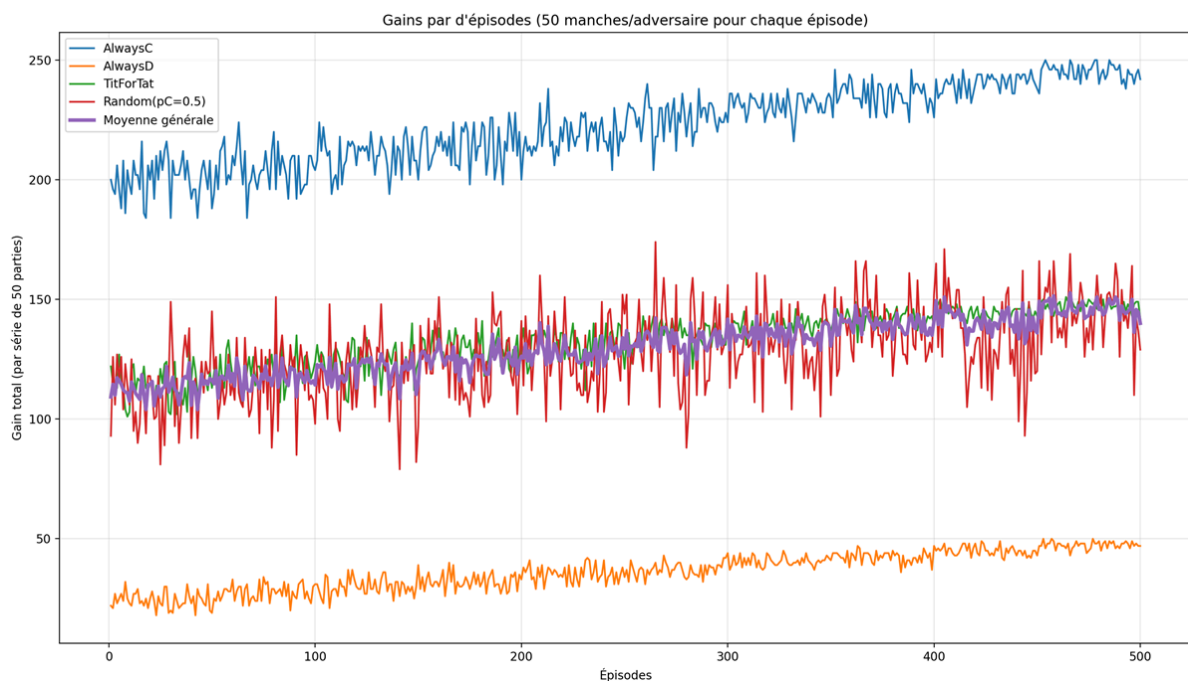
2- Modèle de base

Pour la première étape, notre code fonctionne de la manière suivante :

- 1) Définition de la matrice de gain
- 2) Définition des classes des adversaires et de leurs stratégies

- 3) Définition de la mémoire de l'agent et de ses stratégies
- 4) Interaction joueur adversaire avec la fonction `play_match`
- 5) Implémentation de l'évolution des epsilons et de la fonction d'entraînement
- 6) Graphique à obtenir et lancement.

Avec ce code nous obtenons le graphique suivant :



En visualisant le graphique suivant, nous pouvons nous demander : Est-ce que notre agent a bien appris ?

è Oui, car l'espérance de gain est maximisé contre chacun de ses adversaires.

Pour le cas contre l'adversaire AlwaysC, l'agent a maximisé son rendement car quand notre partenaire coopère, on gagne soit 3 soit 5, donc sur 50 matchs par épisodes, le gain se situe entre 150 et 250. On remarque sur le graphique que pour la courbe correspondante en bleu, l'agent a un gain de 200 aux premiers épisodes, soit le milieu de l'intervalle [150 ; 250], pour tendre vers 250 au fur et à mesure des épisodes.

Pour l'adversaire AlwaysD, le gain est maximisé malgré qu'il soit faible car l'agent peut gagner soit 0 soit 1 à chaque partie, donc entre 0 et 50 par épisodes. On remarque sur le graphique que pour la courbe correspondante en orange, l'agent a un gain de 25 au départ, soit le milieu de l'intervalle [0 ; 50], pour tendre vers 50 au fur et à mesure des épisodes.

Pour l'adversaire Tit-For-Tat, l'agent a maximisé son rendement également. En effet, il n'existe pas vraiment d'équilibre à long terme ou l'agent trahirait l'adversaire pour un meilleur gain à long terme car sinon l'adversaire le trahirait et cela baissera ses gains futurs. On remarque que l'agent apprend de l'adversaire car il ne s'enferme pas dans la trahison. En effet, s'il le trahirait toujours, il aurait un gain moyen de 50 par épisode, et si il le trahirait une fois sur 2, il aurait un gain de $(5+0)*50 / 2 = 125$, alors qu'en coopérant toujours il obtient un gain de 150 par épisode ($3*50 = 150$). C'est ce qu'on obtient à la fin du graphique.

Pour l'adversaire Random, on ne peut pas savoir le choix de l'adversaire. L'agent a maximisé son gain car il suffit de faire un comparatif des différentes possibilités. Si l'agent coopère toujours, il a un gain moyen de $(3+0)*50 / 2 = 75$ alors que s'il trahit toujours, il a un gain moyen de $(5+1)*50 / 2 = 150$. Comme $150 > 75$, alors il doit choisir de toujours trahir et on remarque c'est ce qu'il fait sur les derniers épisodes.

3 – Extension du modèle

