ING4 Intelligence Artificielle

Minimisation de regret hypothétique profond et Poker

Louis Loisel (Gr.04), Nicolas Tronson (Gr.02)

Table des matières

Introduction

Présentation

Dans le cadre de nos études en école d'ingénieurs, l'objectif de ce travail est de découvrir, expérimenter et présenter différentes applications de l'apprentissage automatique et de la théorie des jeux, lié à l'intelligence artificielle. Etant deux joueurs de poker amateurs, nous avons choisis de travailler sur le sujet suivant : minimisation de regret hypothétique profond et Poker.

Contexte

Le poker est un jeu à informations imparfaites aussi appelé jeu bayésien. En effet, ce type de jeu cherchent à analyser les situations où l'on ne connait pas parfaitement les données de jeu. Cette situation impose aux joueurs des croyances concernant les autres joueurs, ces croyances sont représentées comme une distribution de probabilité sur toutes les caractéristiques possibles. L'actualisation se fait en utilisant la règle de Bayes.

Bien que l'Intelligence Artificielle ait déjà réussi à battre les humains à d'autres jeux tels que les échecs, le Go et certains jeux vidéo (des jeux suivant des règles prédéfinies et sans hasard), gagner une partie de poker s'est avéré beaucoup plus complexe. Ce jeu requiert de la stratégie, de l'intuition, et un raisonnement basé sur des informations cachées, le nombre de joueurs compliquant encore l'équation. Dans la théorie de l'IA, le poker est classé comme un environnement d'informations imparfaites, ce qui signifie que les joueurs n'ont jamais une image complète du jeu. Mais malgré ces obstacles, l'intelligence artificielle peut maintenant jouer et gagner au poker.

C'est donc fin 2017 que la première IA a pu être mise en place et a pu battre des joueurs professionnels, ce programme porte le nom de Libratus.

Résumé

A travers ce projet nous allons dans un premier temps étudié les différents programmes qui existent déjà dans le cadre de l'intégration de l'intelligence artificielle dans le milieu du poker. Nous mettrons par la suite en évidence les différentes techniques utilisées, et pour finir nous présenterons notre partie pratique dans laquelle nous expliquerons les procédés que nous avons utilisé afin de développer notre propre intelligence artificielle.

Les programmes déjà existants

Libratus

Libratus est donc un programme informatique d'intelligence artificielle pour le poker à deux joueurs. La construction de celui-ci a nécessité 15 millions d'heure/cœur de calcul pour le supercalculateur américain Bridge pour s'entrainer. D'après ses créateurs, le programme n'a pas de stratégies spécifiques mais la calcul grâce à un algorithme adaptatif en utilisant la méthode de la "minimisation du regret hypothétique" alias Counterfactual Regret Minimization (CFR), nous vous présenterons cette méthode dans la suite de cette présentation.

C'est donc fin janvier 2017 que cet algorithme fut mis en place dans un tournoi avec quatre joueurs professionnels dans un système d'équipe de deux et en faisant en sorte que l'élément de hasard dans les distributions soit annulé. Ce match aura duré 20 jours et au total, 120.000 mains ont été jouées. A la fin de la compétition, Libratus sortira avec un bénéfice de plus de 1,7 millions de dollars alors que les joueurs professionnels enregistrent tous une perte. Pour imaginer le niveau du programme, nous pouvons citer l'un des quatre participants de ce tournoi, Dong Kim qui dira « Jusqu'ici, je ne m'étais pas rendu compte à quel point il était bon. Aujourd'hui, j'avais les mêmes sensations qu'en jouant contre un tricheur qui aurait vu mes cartes. Mais Libratus ne trichait pas, il était seulement trop fort ».

Name	Rank	Results (in chips)
Dong Kim	1	-$85,649
Daniel MacAulay	2	-$277,657
Jimmy Chou	3	-$522,857
Jason Les	4	-$880,087
Total:		-$1,766,250

Le programme Libratus s'organise en trois modules distincts, l'abstraction et la recherche d'équilibre, résolution des sous-jeux imbriqués, "l'auto-amélioration".

1) L'abstraction et la recherche d'équilibre permet de construire un plan directeur de la stratégie. Une solution au problème des informations imparfaites consiste simplement à raisonner sur l'ensemble du jeu, plutôt que sur des morceaux de celui-ci. Dans cette approche, une solution est précalculée pour l'ensemble du jeu, éventuellement en utilisant un programme linéaire ou un algorithme itératif comme l'algorithme de minimisation hypothétique du regret cette méthode peut permettre de résoudre des problèmes avec 10^{13} décisions possibles. Le problème était donc ici de pouvoir résoudre un jeu comme le poker version Heads Up No Limit avec 10^{161} décisions possibles. C'est à ce niveau là qu'intervient l'abstraction, elle permet de regrouper les mains similaires et du traité de la même façon. Cette abstraction n'intervient qu'après les deux derniers tours d'une main car les décisions possibles deviennent alors trop nombreuses pour pouvoir toutes les calculer. Après la construction de cette abstraction, le programme apprend en jouant contre lui-même cet apprentissage utilise l'algorithme de minimisation hypothétique du regret, il permet le maintien d'une valeur de regret pour chaque action. Intuitivement, le regret représente à quel point l'IA regrette de ne pas avoir choisi cette action dans le passé. Lorsqu'une décision est rencontrée pendant l'entrainement, l'IA choisit des actions avec un regret plus élevé avec une probabilité plus élevée.

2) En plus de l'abstraction, Libratus s'appuie sur des recherches antérieures sur la résolution de sous-jeux, dans lesquelles une stratégie plus détaillée est calculée pour une partie particulière qui est atteinte pendant le jeu. Libratus présente de nombreuses avancées dans la résolution de sous-jeux qui se sont avérées essentielles pour atteindre les résultats actuels. Les IA précédentes qui utilisaient la résolution de sous-jeux en temps réel ont résolu ce problème en supposant que l'adversaire joue selon la stratégie du plan directeur. Cependant, l'adversaire peut exploiter cette hypothèse en passant simplement à une stratégie différente. Pour cette raison, la technique peut produire des stratégies bien pires que la stratégie du plan directeur et est appelée résolution de sous-jeux non sécurisée. Maintenant que l'abstraction est plus détaillée dans le sous-jeu et que les stratégies du sous-jeux sont mieux présentées, il peut être possible de trouver une amélioration par rapport à la stratégie précédente qui aggrave la situation de l'adversaire quelles que soient les cartes qu'elle détient.

3) Le troisième module de Libratus est l'auto-amélioration. Il améliore la stratégie du plan directeur en arrière-plan. Il remplit les branches manquantes dans l'abstraction du plan et calcule une stratégie de théorie des jeux pour ces branches. En principe tous ces calculs pourraient être fait à l'avance, mais l'arbre du jeu est bien trop grand pour que ça soit faisable. Permet d'essayer de construire un modèle de l'adversaire, trouver des erreurs dans la stratégie de l'adversaire. Cette auto-amélioration est constitué d'un Deep Learning qui durant la nuit analyse les parties jouées durant la journée et permet d'améliorer la stratégie et d'éliminer les imperfections, l'équilibre de Nash est aussi utilisé dans cette partie.

En conclusion, cette première IA de poker qui a pu battre des joueurs professionnels, on peut se demander si certains programmes futurs pourront battre des professionnels dans des parties à plus de deux joueurs et s'il existe finalement une stratégie parfaite pour le poker. Passons maintenant à l'étude du programme Pluribus qui est une descendant du Libratus.

Pluribus

Plus récemment, en 2019, Pluribus, un programme d'intelligence artificielle développé par l'Université Carnegie Mellon, situé en Pennsylvanie aux Etats-Unis, en collaboration avec Facebook AI, généralise les résultats de Libratus au Poker pour l'adapter à plus de deux joueurs.

Ce programme a déjà fait ses preuves ayant vaincu les principaux joueurs professionnels du poker No-Limit Texas Hold'em à six joueurs, étant la forme de poker la plus populaire dans le monde, mais aussi la plus dure à maîtriser.

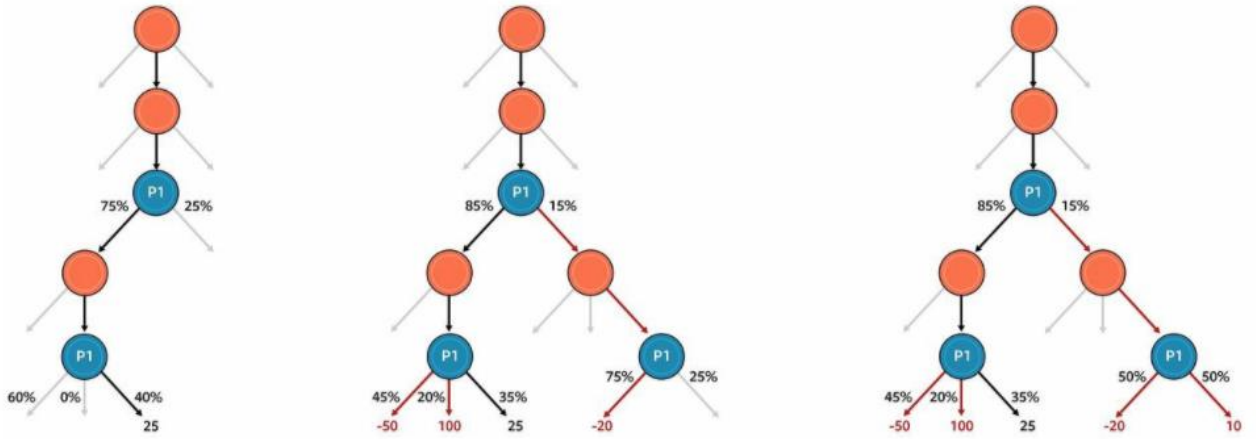
Le défi du poker Texas Hold'em sans limite à six joueurs peut se résumer en trois aspects principaux:

- Traiter des informations incomplètes.
- Difficulté à atteindre un équilibre de Nash (que l'on développera par la suite)
- Le succès nécessite des compétences psychologiques comme le bluff.

Pour relever ce défi, Noam Brown et Thomas Sandholm (deux chercheurs) ont d'abord appris à Pluribus à jouer au poker contre des copies de lui-même, sans qu'aucune donnée de jeu d'IA humaine ou antérieure n'ait été utilisée comme entrée. C'est ce qu'on appelle la méthode du "self-play", une technique couramment employée pour entraîner l'IA. En s'entraînant contre elle-même au travers de milliers de parties, l'IA apprend alors de ses propres erreurs, lui permettant d'évoluer d'elle-même par la suite, en déterminant quelles actions et quelle distribution de probabilité sur ces actions conduisent à de meilleurs résultats par rapport aux versions antérieures de sa stratégie. C'est la méthode de minimisation du regret hypothétique (alias CFR, pour Counterfactual Regret Minimization) qui a été retenue, méthode que l'on développera par la suite.

Cette technique s'est avérée particulièrement efficace. Pour entraîner Pluribus, il aura suffi de huit joueurs et d'un serveur à 64 cœurs équipé de moins de 512 GB de RAM. Le coût de l'opération peut donc être estimé à 150 dollars à peine. Ce qui représente un prix dérisoire dans le but d'humilier les meilleurs joueurs de Poker du monde à leur propre discipline.

Pour automatiser la formation en self-play, l'équipe Pluribus a utilisé une version de l'algorithme itératif Monte Carlo CFR (MCCFR). A chaque itération de l'algorithme, l'algorithme désigne un joueur comme «traverseur» dont la stratégie courante est mise à jour à l'itération. Au début de l'itération, l'algorithme simule une main de poker en fonction de la stratégie actuelle de tous les joueurs (qui est initialement complètement aléatoire). Une fois la main simulée terminée, l'algorithme passe en revue chaque décisions prises par le traverseur et examine à quel point il aurait pu être meilleur ou pire en choisissant les autres actions disponibles à la place. Ensuite, l'IA évalue le gain/perte de chaque décision qui aurait pu être prise à la suite, et ainsi de suite. La différence entre ce que le traverseur aurait reçu pour choisir une action et ce que le traverseur a réellement réalisé sur l'itération s'ajoute au regret contrefactuel pour l'action. À la fin de l'itération, la stratégie du traverseur est mise à jour afin que les actions avec un regret contrefactuel plus élevé soient choisies avec une probabilité plus élevée.



Le Texas Hold'em à six joueurs sans limite possède trop de points de décisions. Pluribus utilise une technique appelée abstraction pour regrouper des actions similaires et en éliminer d'autres réduisant la portée de la décision. La version actuelle de Pluribus utilise deux types d'abstractions :

- **Abstraction d'action :** ce type d'abstraction réduit le nombre d'actions différentes que l'IA doit prendre en compte. Par exemple, parier 150 $ ou 151 $ pourrait ne pas faire de différence du point de vue de la stratégie. Pour équilibrer cela, Pluribus ne considère qu'une poignée de tailles de pari à n'importe quel point de décision.
- **Abstraction d'information :** ce type d'abstraction regroupe les points de décision en fonction des informations qui ont été révélées. Par exemple, une quinte à dix et une quinte à neuf sont des mains distinctes, mais sont néanmoins stratégiquement similaires. Pluribus utilise l'abstraction d'informations uniquement pour raisonner sur des situations lors de futurs tours d'enchères, jamais sur le tour d'enchères dans lequel il se trouve réellement.

De plus, pour permettre à Pluribus de pouvoir affronter 5 joueurs à la fois cette première méthode n'était pas suffisante. Les chercheurs lui ont ensuite appris à décrypter le jeu avec la technique de la "fonction de recherche". Par cette méthode au lieu de prédire ce que vont faire les adversaires jusqu'à la fin de partie, Pluribus est conçu pour penser avec seulement 2 ou 3 tours à l'avance, permettant de réduire la complexité des calculs (sans avoir aucuns effets sur la performance Pluribus ayant fait ses preuves par la suite).

Pour finir la capacité de Pluribus à bluffer (au poker le bluff est une tactique mensongère, visant à faire croire à l'adversaire que l'on a un meilleur jeu que lui) est liée au fait qu'il ne perçoit pas le bluff comme une tromperie mais une méthode d'optimisation mathématique des gains. Pour faire simple, l'IA se contente de prendre la décision qui peut lui rapporter le plus d'argent à chaque tour.

Pour conclure avec Pluribus les résultats ont été à la hauteur de le leurs espérances. Pendant une durée de douze jours, l'IA a affronté 12 professionnels en plus de 10 000 mains. Les parties se sont déroulées dans deux configurations différentes. Dans le premier cas, l'IA jouait seule contre cinq joueurs humains. Dans le second cas, cinq versions de l'IA jouaient contre un humain seul.

Au total, Pluribus a gagné en moyenne 5 dollars par main et environ 1000 dollars par heure. Sa marge de victoire a été qualifiée de "décisive" selon les chercheurs. D'après le chercheur Noam Brown de Facebook AI Research, co-créateur de Pluribus, on peut même parler d'un "niveau surhumain".

Ses adversaires eux-mêmes avouent être bluffés. Selon Chris Ferguson, six fois champion du World Series of Poker, "Pluribus est très difficile à affronter, quelle que soit votre main". Les joueurs ont été choqués par sa capacité à bluffer ses adversaires, et à gagner des parties même avec des mains médiocres. C'est la constance de l'IA qui a surtout surpris les professionnels.

Dans un article publié par "Science", les scientifiques expliquent que la victoire de Pluribus au poker est une avancée majeure pour l'IA.

Les techniques utilisées

L'équilibre de Nash

En théorie des jeux, l'équilibre de Nash, du nom du mathématicien John Forbes Nash Jr., est le moyen le plus courant de définir la solution d'un jeu non coopératif impliquant deux joueurs ou plus. Dans un équilibre de Nash, chaque joueur est supposé connaître les stratégies des autres joueurs et aucun joueur n'a rien à gagner en ne changeant que sa propre stratégie. Illustrons cet équilibre grâce à un problème connu appelé les marchands de glaces. Sur une plage, 2 marchands sont présents et ceux-ci ont pour objectifs d'avoir un maximum de clients. Les deux marchands sont placés de manière équidistante du centre de la plage, chacun d'un côté, dans cette situation-là, les deux marchands ont les mêmes résultats comptables. Supposons que l'un des marchands décide de se rapprocher du centre de la place pour agrandir son champ d'action, le seul moyen pour le second de récupérer ses clients est pour lui aussi se rapprocher du centre. Cette situation se répète jusqu'à ce que les deux marchands se retrouvent au milieu de la plage et auront finalement, comme au début, les mêmes résultats. Cette situation est appelée en théorie des jeux, l'équilibre de Nash. Cette dernière situation est dites non-optimale, en effet si on suppose que les clients se dirigent vers le marchand le plus proche, il n'est pas dit que certain n'ai pas envie de marcher des centaines de mètre pour obtenir leurs glaces. Les deux marchands verront alors leurs chiffres d'affaires diminuer. C'est donc la situation de départ qui serait la plus favorable mais elle ne pourra être retrouver sans un accord des deux contre parties.

Au poker, cet équilibre de Nash est utilisé pour créer des arbres de décisions pour chaque partie. Or cet arbre de décision est très grand est presque impossible de résoudre de façon mathématique même grâce aux superordinateurs. Comme nous l'avons vu précédemment, les IA qui ont battu des joueurs professionnels ont utiliser le système d'abstraction pour « approximer » le jeu.

Remarque : un équilibre de Nash dans un sens n'est pas une stratégie optimale au poker. En effet, les IA de poker actuelles traitent chaque main comme un jeu séparé. Au lieu de cela, un équilibre de Nash définit une limite inférieure sur ce que vous pouvez perdre. C'est une stratégie défensive. Deux joueurs suivant des stratégies d'équilibre de Nash perdraient de l'argent au fil du temps puisque le poker n'est pas vraiment un jeu à somme nulle.

Minimisation de regret hypothétique (CFR)

La minimisation contrefactuelle des regrets est un algorithme qui se rapproche d'un équilibre de Nash. L'algorithme du CFR minimise les regrets sur de nombreuses itérations jusqu'à ce que la stratégie moyenne sur toutes les itérations converge.

La méthode du CFR peut être expliquée en quelques points clés :

- C'est un algorithme d'entraînement avec soi-même : l'IA apprend en jouant contre elle-même
- On appelle regret ce qu'on aurait gagné à jouer telle action fixée à tel moment pour toutes les parties précédentes, par rapport à ce qu'on a fait à la place.
- Au départ, la stratégie est aléatoire, mais après chaque parties toutes les décisions prises sont revues grâce aux regrets :
    - Si le regret est positif, il faudra changer plus souvent l'action pour celle calculée
    - S'il est négatif, alors on a fait la meilleure chose possible et il faut continuer ainsi

Notre stratégie est ainsi revue pour qu'elle tienne compte des regrets positifs (la probabilité de faire une action dépend donc de son intérêt pour nous faire gagner globalement). Autrement dit, on ne cherche pas à gagner une partie mais le maximum de parties.

L'algorithme ne converge pas toujours, sauf dans le cas du poker où il est possible d'atteindre ce qu'on appelle "l'équilibre de Nash", qui est une stratégie gagnante si elle est parfaitement respectée puisque, sur un grand nombre de parties, on obtient qu'en moyenne :

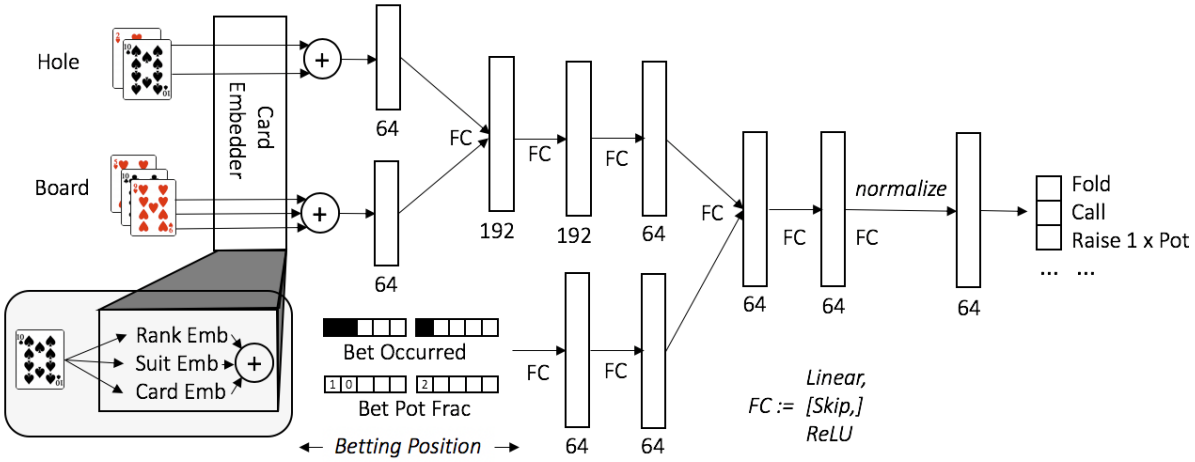
- Si l'adversaire suit une stratégie de Nash, on finira à égalité
- S'il joue une stratégie parfaite qui contre Nash, on finira aussi à égalité
- Sinon, à la moindre erreur, je gagnerai

On ne gagne donc que par la meilleure défense qui soit (en exploitant les erreurs de l'adversaire pour marquer des points).

Nous allons maintenant voir comment avec le Deep Learning les chercheurs ont pu améliorer les précédentes versions du CFR.

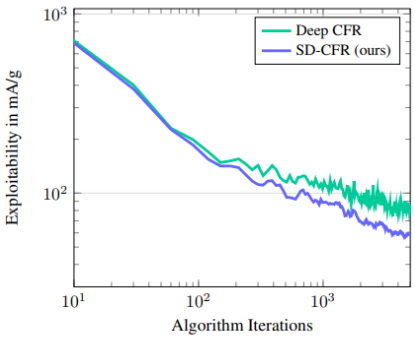
Deep CFR

Comme nous avons pu le voir dans les parties précédentes, CFR est le principal algorithme de résolution des jeux à informations imparfaites. Il parcourt de manière itérative l'arbre du jeu afin de converger vers un équilibre de Nash. Afin de gérer des jeux extrêmement volumineux, CFR utilise généralement un processus appelé abstraction comme vu pour le Libratus. Le Deep CFR, une forme de CFR qui évite le besoin d'abstraction en utilisant à la place une série de deux réseaux de neurones profonds pour se rapprocher du comportement de CFR dans le jeu sans abstraction. Le but de Deep CFR est d'approximer le comportement du CFR tout en évitant de traverser complètement l'arborescence du jeu. L'architecture neuronales utilisée pour le calcul des valeurs de jeu et des actions de chaque joueur est le suivant,

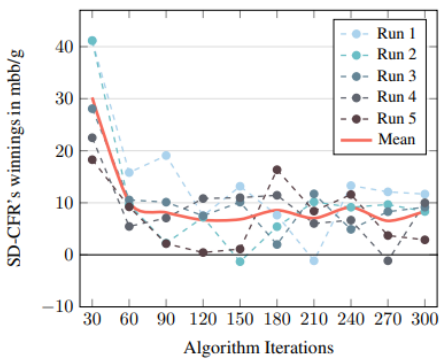


Celle-ci est composée de 7 couches de neurones ainsi que de 61.604 paramètres.

Plus récemment encore, des améliorations ont été proposé notamment par Eric Steinberger un étudiant de l'université de Cambridge. Celui-ci à introduit l'idée du Single Deep CFR à travers une publication scientifique. Dans cet article, la série de deux approximations neuronales est remplacé par une seule approximation neuronale, d'où son nom Single Deep CFR. Cette méthode permettrait de réduit l'erreur globale d'échantillonnage et d'approximation et rend l'entraînement plus efficace. L'article démontre aussi que cette nouvelle approche surclasse le Deep CFR dans les parties en 1 contre 1, comme illustré ci-dessous.



(a) Comparing SD-CFR and Deep CFR



Le Deep CFR et sa variante le Single Deep CFR seront probablement l'une des solutions pour trouver l'équilibre de Nash au poker.

Partie pratique

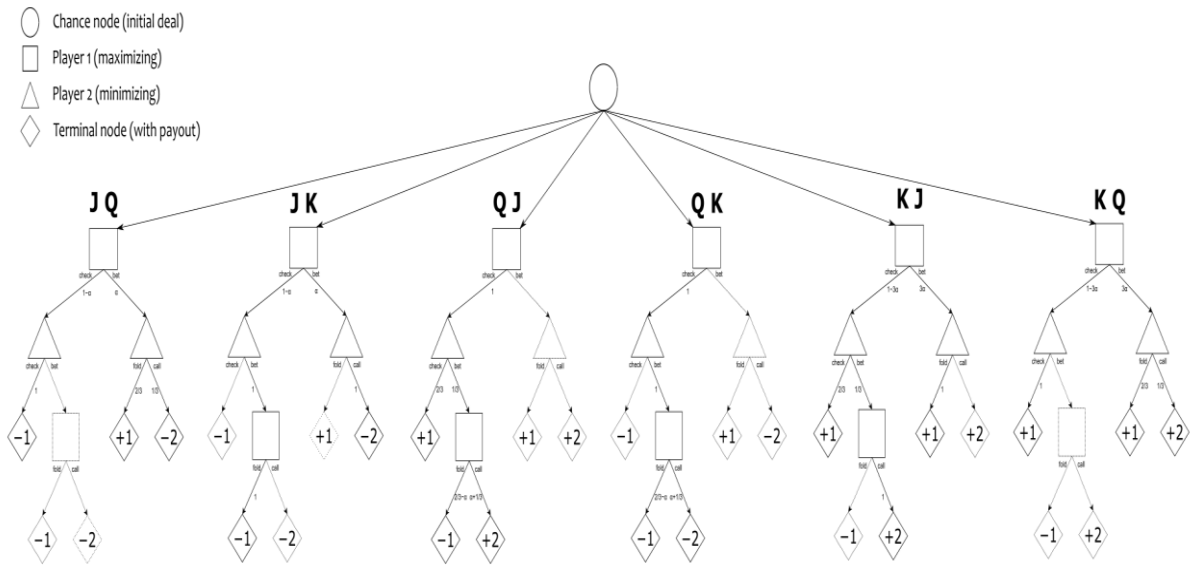
Le poker Kuhn

Il est plus facile de voir le CFR en action en implémentant un algorithme CFR sur une forme plus simple de poker. Nous avons choisi de l'appliquer sur le poker Kuhn.

Le poker Kuhn est une forme extrêmement simplifiée de poker développée par Harold W. Kuhn comme un simple modèle de jeu à deux joueurs, qui se prête à une analyse complète de la théorie des jeux. Au poker Kuhn, le jeu ne comprend que trois cartes. Une carte est distribuée à chaque joueur, qui peut placer des paris de la même manière qu'un poker standard. Si les deux joueurs parient ou que les deux joueurs passent, le joueur avec la carte la plus élevée gagne, sinon, le joueur qui parie gagne. Kuhn a démontré qu'il existe de nombreuses stratégies optimales pour le premier joueur, mais une seule pour le deuxième joueur, et que, lorsqu'elles sont jouées de manière optimale, le premier joueur devrait s'attendre à perdre à un taux de -1/18 par main.

On comprend donc que le joueur un doit parier avec une probabilité égale à 3*alpha avec alpha compris entre [0 ;1/3] lorsqu'il à la carte la plus haute parmi les trois présentent dans le jeu et si l'adversaire suit, il doit toujours le suivre. En revanche, lorsqu'il a la carte intermédiaire, il doit toujours checker puis si l'adversaire suit, il suit la probabilité alpha + 1/3. Avec la carte la plus basse, il choisit librement une probabilité dans le alpha, si le second joueur suit il devra alors obligatoirement se coucher.

Le deuxième joueur quant à lui, à une stratégie d'équilibre unique, miser ou suivre quand il a la carte la plus haute, avec la carte intermédiaire suivre une probabilité de 1/3 et lorsqu'il à la carte la plus basse, ne jamais suivre et miser avec une probabilité de 1/3. L'imagine suivante nous illustre l'arbre complet du poker de Kuhn avec les probabilités d'équilibre de Nash. (Zoomer pour mieux voir)



Explication du code

Chaque itération de cfr() renvoie la valeur de jeu attendue pour cette itération. Pour estimer la valeur réelle du jeu, il faut faire la moyenne de toutes les itérations. La vraie valeur du jeu est le montant attendu qu'un joueur gagnera tout en suivant la stratégie moyenne.

La méthode cfr() contient l'essentiel de la logique. La fonction effectue de manière récursive une traversée en profondeur dans l'arborescence du jeu. cfr() effectue différentes tâches selon que le nœud est un nœud aléatoire, un nœud terminal ou un nœud de décision.

- history est une chaîne qui représente notre emplacement actuel dans l'arborescence du jeu.
- pr_1 est la contribution du joueur 1 pour arriver à la probabilité
- pr_2 est la contribution du joueur 2 pour arriver à la probabilité.

is_terminal() vérifie si l'historique fait partie de l'ensemble prédéfini d'historiques de terminaux connus. Si Vrai, get_reward() retourne le gain du joueur actuel. Puisque les joueurs alternent leurs tours, le joueur actuel est le joueur 1 si le nombre d'actions dans l'histoire est pair et le joueur 2 sinon. Pour calculer le gain d'un historique terminal, il y a trois cas. Si le dernier joueur s'est couché, le joueur actuel gagne 1 jeton. Si les deux joueurs ont checké pendant le tour d'enchères, les joueurs s'affrontent avec un pot de 2. Le joueur avec la carte la plus élevée gagne 1 jeton. Les joueurs s'affrontent avec un pot de 4. Le joueur avec la carte la plus élevée gagne 2 jetons.

Get node génère la clé et récupère le jeu d'informations. Si l'ensemble d'informations n'est pas déjà dans la carte, une nouvelle instance est créée et insérée.

display_results() affiche les ensembles d'informations de chaque joueur, affiche la clé du jeu d'informations en utilisant self.get_average_strategy() pour afficher la stratégie moyenne. self.average_strategy() calcule la stratégie moyenne à partir de self.strategy_sum et définit les probabilités proches de zéro à zéro.

Voici les résultats obtenus après l'utilisation de ce programme au bout de 100 000 itérations.

```
player 1 expected value: -0.05752157608504683
player 2 expected value: 0.0575215760850468 3

player 1 strategies:
0       ['0.74', '0.26']
0 pb    ['1.00', '0.00']
1       ['0.99', '0.01']
1 pb    ['0.40', '0.60']
2       ['0.22', '0.78']
2 pb    ['0.00', '1.00']

player 2 strategies:
0 b     ['1.00', '0.00']
0 p     ['0.67', '0.33']
1 b     ['0.64', '0.36']
1 p     ['1.00', '0.00']
2 b     ['0.00', '1.00']
2 p     ['0.00', '1.00']
```

Ce résultat démontre bien que le second joueur n'a qu'une seule stratégie d'équilibre comme expliqué dans les paragraphes précédents. Pour le joueur 1, on peut observer la stratégie d'équilibre obtenu par l'entrainement grâce au CFR avec une espérance de gain par main de -0.0575 qui correspond bien au résultat obtenu par Harold W.Kuhn de -1/18.

Les stratégies indiquent que si le joueur 1 a un 0, son premier coup devrait être de checker 74% du temps et de miser 26% du temps, par la suite si le joueur 2 bet le joueur 1 va se coucher 100% du temps (ligne 2 de la stratégie du joueur 1, 0 pb) Si le joueur 2 a un 2 et le joueur 1 check, il doit miser 100% du temps. C'est juste un équilibre de Nash. Le poker de Kuhn en a une infinité.

Le second programme que nous avons choisi est le KuhnPoker-master. Celui-ci s'entraine afin de trouver la stratégie optimale pour le joueur 1. Cette stratégie sera alors utilisée pour jouer contre l'utilisateur, l'utilisateur sera donc le joueur numéro 2 et le programme utilisera la stratégie calculer pour le joueur numéro 1. Nous avons rajouté ce programme si vous souhaitez jouer contre l'intelligence artificielle et tester l'efficacité de l'algorithme CFR.

Conclusion

Au-delà de la correction infligée aux meilleurs joueurs de Poker, on peut se demander de quelle façon cette avancée pourra être appliquée à d'autres domaines. En effet, de nombreuses situations du monde réel ressemblent au Poker Texas Hold'em. Des situations impliquant plusieurs personnes, des informations cachées ou incomplètes, et de multiples possibilités. Ainsi, ces premières avancées réalisées par l'intelligence artificielle dans le domaine du poker sont particulièrement prometteuses, les chercheurs espèrent appliquer leurs travaux à des secteurs comme la cybersécurité, la prévention de fraude, et les négociations financières. Ces applications à de nouveaux domaines pourrait représenter une réelle révolution, et une nouvelle manière d'aborder les problèmes.

Notre projet représente une recherche sur les différentes technologies actuelles liées à l'intelligence artificielle dans le domaine du poker. Nous avons tous les deux particulièrement apprécié réaliser ce projet. La réalisation de cette étude était très intéressante, nous avons découvert de nombreuses choses que nous ne connaissions pas encore même si nous sommes tous deux des joueurs de poker occasionnels. De plus se plonger et comprendre comment réaliser un code de ce type était très enrichissant.

Bibliographie

CMU. (2019, 07 11). *Carnegie Mellon and Facebook AI Beats Professionals in Six-Player Poker*. Retrieved from https://www.cmu.edu/news/stories/archives/2019/july/cmu-facebook-ai-beats-poker-pros.html

Int8. (2018, 09 23). *Counterfactual Regret Minimization – the core of Poker AI beating professional players*. Retrieved from https://int8.io/counterfactual-regret-minimization-for-poker-ai/

Juxiann. (n.d.). *KuhnPoker*. Retrieved from https://github.com/Juxiann/KuhnPoker

Martin Zinkevich, M. J. (n.d.). *Regret Minimization in Games with Incomplete.* Retrieved from https://poker.cs.ualberta.ca/publications/NIPS07-cfr.pdf

Noam Brown, A. L. (2019, 05 22). *Deep Counterfactual Regret Minimization.* Retrieved from https://arxiv.org/pdf/1811.00164.pdf

Noam Brown, T. S. (2018, 01 26). *Superhuman AI for heads-up no-limit poker: Libratus beats top professionals*. Retrieved from https://science.sciencemag.org/content/359/6374/418

R., L. (n.d.). *Une IA championne de poker ?* Retrieved from https://penseeartificielle.fr/ia-poker-libratus-bat-professionnels/

Sermeno, J. (n.d.). *Vanilla Counterfactual Regret Minimization for Engineers*. Retrieved from https://justinsermeno.com/posts/cfr/?fbclid=IwAR3HCRDPR-yXGQ1Q6ARs-K8uX75mB2gnK7nuDE7y05H7e-lQCba_4rf-Ftw

Steinberger, E. (2019, 10 04). *Single Deep Counterfactual Regret Minimization.* Retrieved from https://arxiv.org/pdf/1901.07621.pdf

Todd W. Neller, M. L. (2013, 07 9). *An Introduction to Counterfactual Regret Minimization.* Retrieved from http://modelai.gettysburg.edu/2013/cfr/cfr.pdf

Wikipedia. (n.d.). *Libratus*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Libratus