BlackJack : Comment battre le croupier ?
GitHub Repository

Vincent Gardies - Frédéric Becerril

December 11, 2024

Introduction

Contexte et objectifs

Le projet de ce rapport s'inscrit dans le cadre de l'étude des processus de décision markoviens (PDM) appliqués à un jeu de Blackjack, où l'objectif principal est de développer des stratégies optimales pour le joueur et le croupier. En adoptant un modèle markovien, nous analysons les meilleures actions possibles pour maximiser le gain dans différents scénarios et sous diverses hypothèses.

Motivation

Les jeux de stratégie comme le Blackjack constituent un domaine de recherche intéressant pour l'application des processus de décision markoviens. Ils permettent non seulement d'analyser et de comprendre les stratégies optimales dans des contextes stochastiques, mais aussi d'explorer l'utilisation de modèles d'apprentissage par renforcement pour des décisions adaptatives et robustes. Ce projet s'intéresse à une approche progressive de l'optimisation, allant d'une modélisation simplifiée avec des hypothèses de tirage indépendant identiquement distribué (i.i.d.), jusqu'à une modélisation réaliste du tirage sans remise, en passant par des méthodes d'apprentissage supervisé et non supervisé.

Méthodologie

Nous avons structuré notre approche en trois grandes parties. Dans la première, nous commençons par calculer les politiques optimales pour le croupier et le joueur dans un cadre simplifié, avant d'inclure progressivement des règles plus avancées du Blackjack. La deuxième partie s'intéresse aux stratégies optimales dans un contexte de tirage sans remise, avec un nombre fini de cartes pour chaque valeur. Enfin, dans la troisième partie, nous appliquons des techniques d'apprentissage par renforcement, où nous introduisons un modèle de décision supervisé basé sur les résultats de calculs optimaux précédents, puis un modèle non supervisé capable de s'évaluer de manière autonome, et finalement un modèle d'optimisation de la mise en fonction de l'état du paquet de cartes.

Structure du rapport

Le rapport est structuré de manière progressive pour présenter nos analyses et résultats pour chaque partie. Dans chaque section, nous décrivons les méthodes utilisées, les résultats obtenus, et les conclusions tirées, avec une attention particulière portée aux algorithmes employés et à la méthodologie d'expérimentation. Enfin, une conclusion récapitule les principaux résultats et propose des pistes d'amélioration ainsi que des perspectives pour des applications futures.

1 Calcul des Politiques Optimales du Croupier et du Joueur

1.1 Première approche simplifiée

1.1.1 Hypothèses

Dans un premier temps, nous avons adopté un modèle simplifié du Blackjack avec les hypothèses suivantes :

- Valeur d'un As fixée à 1.
- Tirage avec remise, chaque carte ayant une probabilité uniforme d'être tirée.
- Actions possibles pour le joueur et le croupier : `HIT` ou `STAND`.

- Pas de règle de Blackjack naturel (21 dès le premier tirage).

Ces hypothèses permettent de restreindre la complexité du problème à des états définis par les scores des joueurs uniquement, facilitant ainsi le calcul des stratégies optimales.

1.1.2 Méthode de Résolution

Dans ce cadre simplifié, les connaissances respectives du joueur et du croupier se limitent aux éléments suivants :

- **Joueur** : connaît son score (`ScorePlayer`) et la première carte du croupier (`DealerCardValue`), définissant 210 états possibles (21×10).
- **Croupier** : connaît son propre score (`ScoreDealer`) et celui du joueur (`PlayerScore`), définissant 441 états possibles (21×21).

1.1.3 Calcul de la Politique Optimale du Croupier

En exploitant la connaissance du score du joueur, le croupier adapte sa politique selon les cas suivants :

Cas	Action	Résultat
ScorePlayer > 21	Partie terminée	Croupier gagne
ScoreDealer > 21	Partie terminée	Joueur gagne
ScorePlayer $<$ ScoreDealer	STAND	Croupier gagne
ScorePlayer $>$ ScoreDealer	HIT	Indéterminé
ScorePlayer $=$ ScoreDealer	?	Indéterminé

Dans le cas d'égalité, le croupier choisit `HIT` si la probabilité de tirer une carte sécurisée dépasse la probabilité de dépassement :

$$\begin{cases} \text{HIT} & \text{si } \texttt{probaHitSafe} > \texttt{probaBust} \\ \text{STAND} & \text{sinon} \end{cases}$$

1.1.4 Résultats

La Figure 1 présente la politique optimale du croupier pour cette approche simplifiée, et la Figure 2 montre le gain associé.

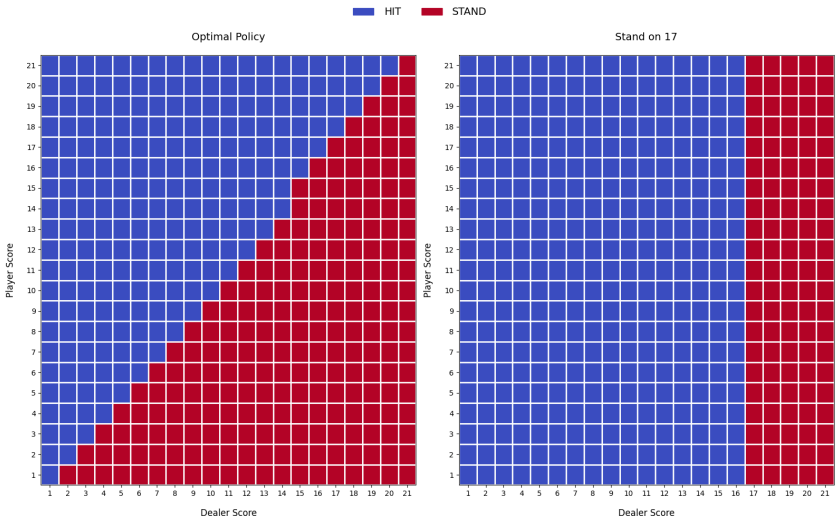

Figure 1: Politique optimale du croupier (approche simplifiée)

1.1.5 Calcul du Gain et de la Politique Optimale du Joueur

Nous avons calculé la politique optimale du joueur en utilisant une programmation dynamique définie comme suit :

$$\text{Gain}[\text{CardDealer}, \text{ScorePlayer}] = \max \begin{cases} \text{HitGain}[\text{CardDealer, ScorePlayer}] & \Rightarrow \text{HIT} \\ -\text{DealerGain}[\text{ScoreDealer, ScorePlayer}] & \Rightarrow \text{STAND} \end{cases}$$

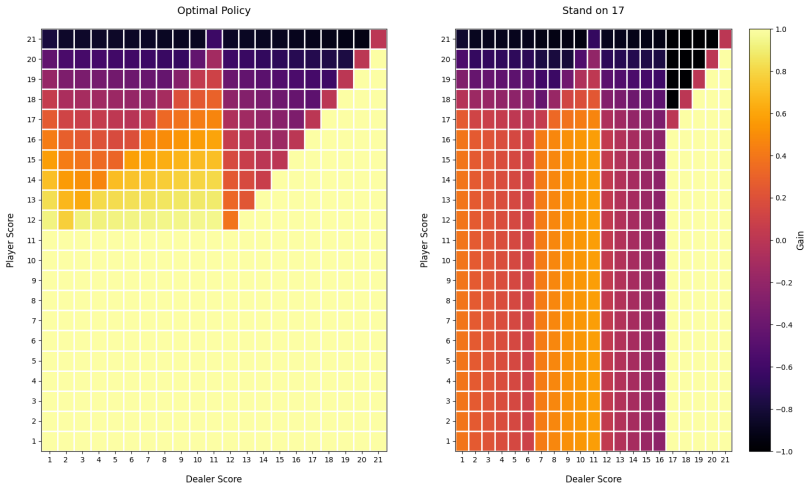

Figure 2: Gain du croupier (approche simplifiée)

avec :

$$\text{HitGain[CardDealer, ScorePlayer]} =$$

$$-\texttt{probaBust} + \sum_{\text{Card}} \texttt{Gain[ScoreDealer, ScorePlayer + Card]} \times \texttt{ProbaDraw(Card)}$$

La Figure 3 présente la politique optimale du joueur, tandis que la Figure 4 montre son gain moyen.

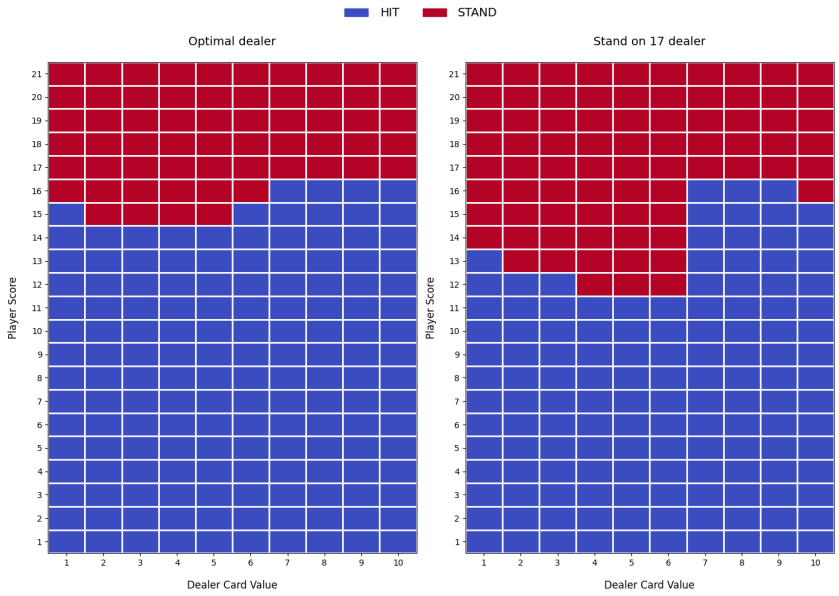

Figure 3: Politique optimale du joueur (approche simplifiée)

1.2 Approche Avancée : Prise en Compte de Règles Additionnelles

1.2.1 Hypothèses supplémentaires

Dans cette seconde approche, nous élargissons le cadre de modélisation en intégrant plusieurs règles traditionnelles du Blackjack pour rendre la stratégie plus réaliste. Les hypothèses ajoutées incluent :

- La valeur de l'As peut être 1 ou 11, selon le contexte.
- La possibilité d'obtenir un Blackjack naturel (21 avec les deux premières cartes).
- Les actions supplémentaires pour le joueur : `SPLIT`, `DOUBLE`, et `SURRENDER`.

Bien que le tirage reste uniforme et avec remise, ces modifications augmentent considérablement l'espace des états, nécessitant une prise en compte détaillée des mains.

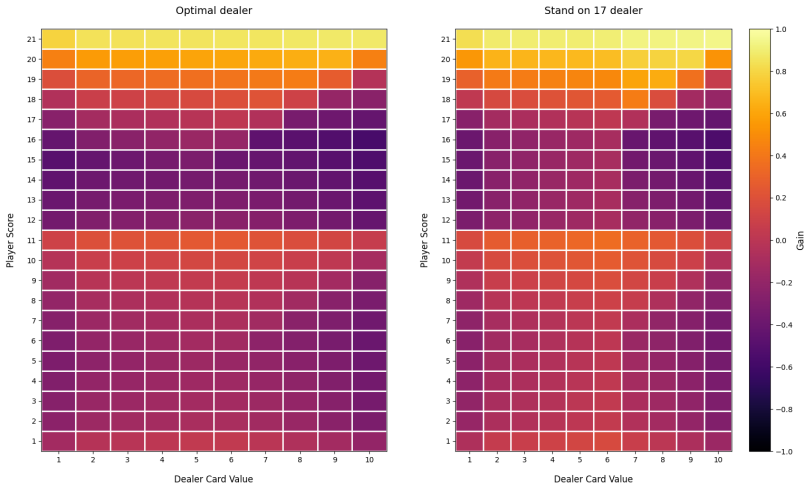

Figure 4: Gain moyen du joueur (approche simplifiée)

1.2.2 Choix d'implémentation

Pour gérer les nouvelles règles, nous modélisons les mains sous forme de tuples, où chaque carte possède sa propre valeur. Cela nous permet de suivre précisément les combinaisons de cartes pour chaque joueur :

- `Card = int`
- `Hand = tuple[int, int, ..., int]` (représentant la composition des cartes de la main)
- `Policy = Callable[[Hand, Hand], Action]` (politique de décision pour chaque joueur en fonction des mains)

Avec ces représentations, nous pouvons aisément intégrer des actions comme le `SPLIT` et le `DOUBLE`, tout en augmentant la complexité de l'espace des états.

1.2.3 Méthode de Résolution

Dans cette configuration avancée, l'espace des états est défini de la manière suivante :

- **Joueur** : connaît sa main détaillée (`HandPlayer`) et la première carte du croupier (`DealerCard`), ce qui produit un espace d'environ $47,801$ états (3677×13).
- **Croupier** : connaît sa propre main (`HandDealer`) et la main du joueur (`HandPlayer`), produisant un espace d'environ $13,520,329$ états (3677×3677).

1.2.4 Politique et Gain du Croupier

La politique optimale du croupier est calculée par programmation dynamique, avec des décisions prises en fonction des nouvelles actions possibles :

$$\text{Gain}[\text{HandDealer}, \text{HandPlayer}] = \max \begin{cases} \text{HitGain}[\text{HandDealer, HandPlayer}] & \Rightarrow \text{HIT} \\ \text{ResultGame}[\text{HandDealer, HandPlayer}] & \Rightarrow \text{STAND} \end{cases}$$

où :

$$\text{HitGain}[\text{HandDealer, HandPlayer}] = \sum_{\text{Card}} \text{Gain}[\text{HandDealer} + \text{Card, HandPlayer}] \times \text{ProbaDraw}(\text{Card})$$

et `Gain[InvalidHand] = -1` en cas de dépassement.

1.2.5 Politique et Gain Optimaux du Joueur

Avec les nouvelles règles, la politique optimale du joueur devient plus complexe, en intégrant des actions telles que le `SPLIT`, le `DOUBLE`, et le `SURRENDER` :

$$\text{Gain}[\text{HandDealer}, \text{HandPlayer}] = \max \begin{cases} \text{HitGain}[\text{HandDealer, HandPlayer}] & \Rightarrow \text{HIT} \\ -\text{GainDealer}[\text{HandDealer, HandPlayer}] & \Rightarrow \text{STAND} \end{cases}$$

où :

$$\text{HitGain}[\text{HandDealer, HandPlayer}] = \sum_{\text{Card}} \text{Gain}[\text{HandDealer, HandPlayer + Card}] \times \text{ProbaDraw}(\text{Card})$$

Les Figures 5 et 6 montrent respectivement la politique optimale du joueur et son gain moyen en intégrant ces nouvelles règles.

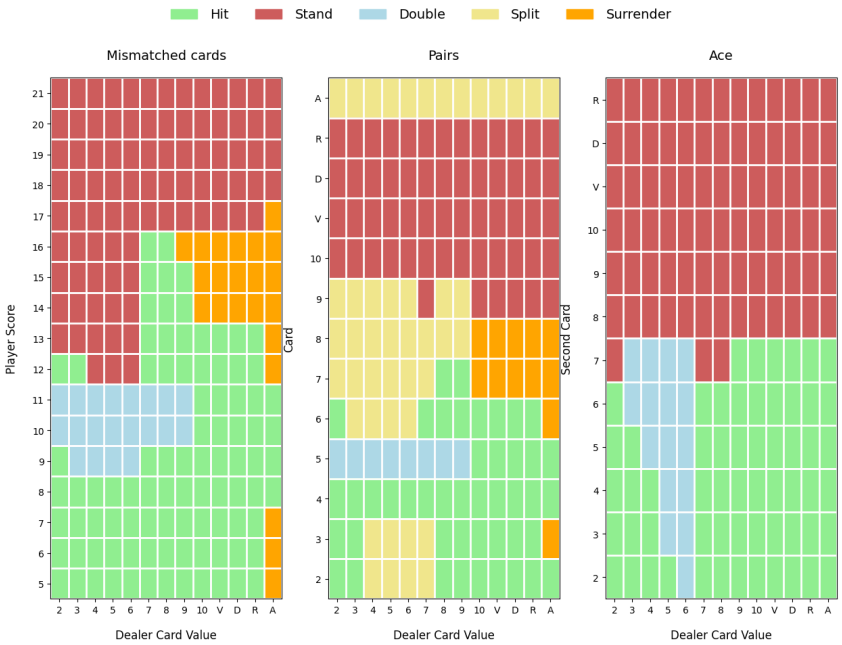

Figure 5: Politique optimale du joueur avec règles avancées

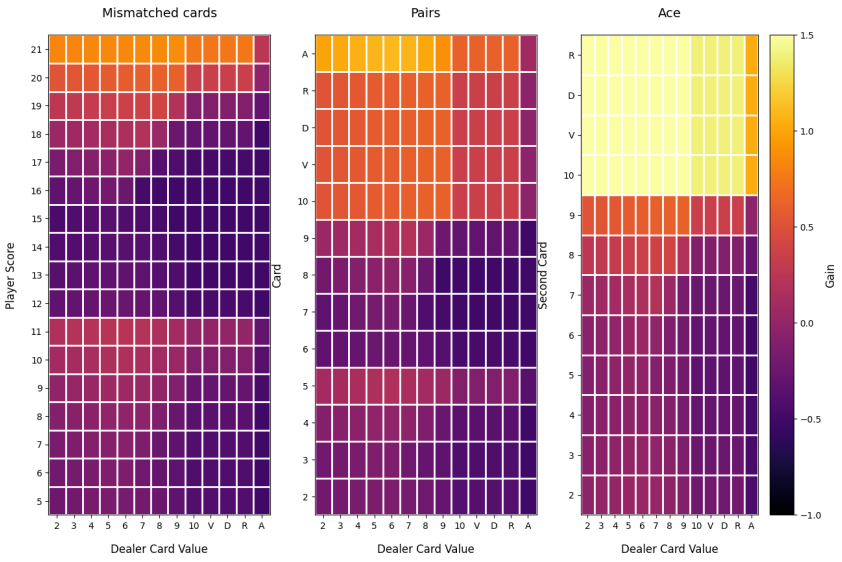

Figure 6: Gain moyen du joueur avec règles avancées

1.2.6 Comparaison avec Wikipédia

Sur la page française du Blackjack de Wikipédia, une stratégie dite "de base" est décrite. Cette stratégie dépend notamment des règles et du nombre de jeux de cartes utilisés. Elle est illustrée à la Figure 7.
Nous avons comparé notre stratégie optimale avec celle de Wikipédia, qui diffèrent en plusieurs cas. Les résultats de cette comparaison sont présentés dans le Tableau 1.
Pour les mêmes actions possibles, notre stratégie optimale génère un gain moyen supérieur à celui de la stratégie décrite sur Wikipédia, quelle que soit la stratégie du croupier parmi celles évoquées. Ce résultat valide notre approche, bien qu'il faille noter que les paramètres utilisés pour la stratégie Wikipédia ne sont pas entièrement explicites.

Figure 7: Stratégie de base selon Wikipédia

	17Soft	**17Hard**	**Gain Total**
Stratégie Optimale	-0.077	-0.075	-0.164
Stratégie Wikipédia	-0.104	-0.102	-0.214

Table 1: Comparaison des gains entre notre stratégie et celle de Wikipédia

2 Partie II : Analyse sous l'hypothèse de tirage sans remise

2.1 Hypothèses et Modélisation

Dans cette partie, nous modifions le cadre du jeu en adoptant une hypothèse de tirage sans remise, ce qui implique un nombre fini de cartes par valeur. Contrairement au tirage avec remise, la probabilité de tirer une carte spécifique change en fonction des cartes déjà tirées, ce qui ajoute un niveau de complexité et permet au joueur d'ajuster ses décisions en conséquence.
Les hypothèses spécifiques de cette partie incluent :

- Il existe un nombre fixé N de cartes pour chaque valeur (par exemple, $N = 4$ pour un deck complet de Blackjack avec 4 occurrences de chaque carte).
- Le joueur connaît la composition du paquet de cartes restant, mais pas le croupier, qui continue de jouer selon une politique fixe.

2.2 Adaptation de la Méthodologie

Avec la nouvelle hypothèse de tirage sans remise, l'espace des états évolue pour inclure l'état du paquet de cartes. Pour chaque main jouée, la probabilité de tirer une carte spécifique dépend de la composition restante du paquet, ce qui nécessite une approche plus sophistiquée pour calculer les probabilités et les politiques optimales.
La méthode employée repose sur une programmation dynamique qui prend en compte :

- La main actuelle du joueur et celle du croupier.
- L'état actuel du deck (composition des cartes restantes).
- Les probabilités de tirage qui varient en fonction des cartes restantes.

2.3 Calcul de la Politique Optimale du Joueur

Étant donné que le croupier utilise la politique optimale calculée précédemment, le joueur peut ajuster sa stratégie en fonction de l'état du paquet. Nous avons modélisé la politique optimale du joueur de la

manière suivante :

$$\text{Gain[HandD, HandP, Deck]} = \max \begin{cases} \text{HitGain[HandD, HandP, Deck]} & \Rightarrow \text{HIT} \\ -\text{GainDealer[HandD, HandP, DeckUpdate]} & \Rightarrow \text{STAND} \end{cases}$$

où :

$$\text{HitGain[HandD, HandP, Deck]}$$
$$= \sum_{\text{Card}} \text{Gain[HandDealer, HandPlayer + Card, DeckUpdate]} \times \text{ProbaDraw(Card | Deck)}$$

et `DeckUpdate` représente le paquet de cartes mis à jour après le tirage de la carte.

2.4 Résultats

Dans un premier temps, on s'intéresse au gain moyen pour des jeux de cartes **homogènes** de différentes tailles. On obtient le tableau 2

	17Soft	**17Hard**	**Optimale**
1 carte de chaque	-0.123	-0.121	-0.205
2 cartes de chaque	-0.107	-0.105	-0.184
3 cartes de chaque	-0.102	-0.100	-0.178
1 jeu de cartes	-0.099	-0.097	-0.174
2 jeux de cartes	-0.095	-0.092	-0.170
7 jeux de cartes	-0.092	-0.089	-0.166

Table 2: Gain global pour différents decks en fonction de la politique du dealer

On observe d'une part que, en augmentant le nombre de cartes de manière homogène, on se rapproche du gain optimal sans comptage de cartes, ce qui est cohérent. Une seconde observation est que, peu importe la politique, avec un jeu homogène, le gain moyen pour le joueur augmente avec le nombre de cartes. Cela peut paraître contre-intuitif étant donné que le joueur devrait connaître des informations "plus importantes" sur la distribution du jeu.

On peut aussi s'intéresser au gain moyen sur des jeux aléatoires. Les jeux sur lesquels le gain a été calculé ont un nombre aléatoire et indépendant de chaque carte, entre 0 et 28, soit 7 jeux complets. Les résultats obtenus sont dans la figure 8 :

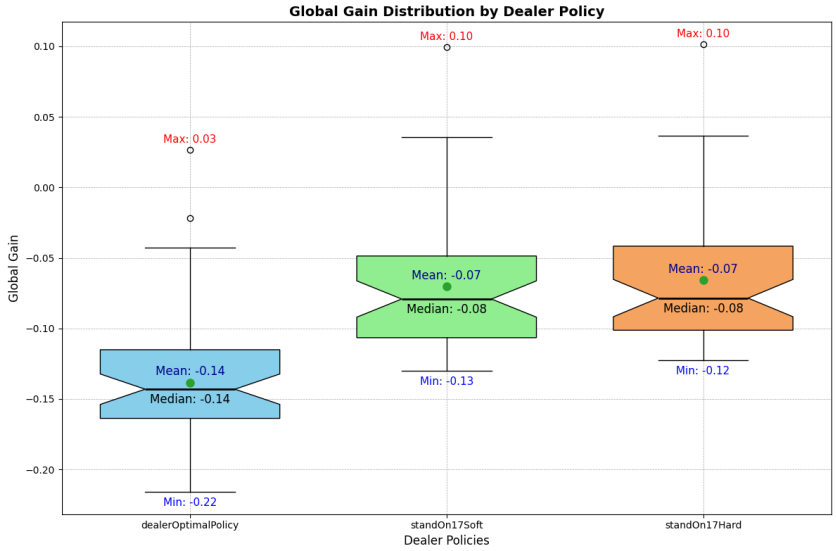

Figure 8: Gain moyen du joueur pour des decks aléatoires dans un jeu de 7 paquets

Sur ces graphiques, on observe la variation élevée du gain moyen selon le deck. On remarque qu'une petite proportion de decks permet des gains globaux positifs pour les politiques Stand, et une proportion encore plus petite pour la politique optimale (1 deck sur 50 dans nos résultats). Cependant, malgré la diversité des decks, le joueur a tendance à augmenter son gain en comptant les cartes, ce qui est cohérent.

Enfin, on peut s'intéresser à des jeux plus probables de survenir lors d'une partie de Blackjack. Pour cela, nous avons simulé des decks auxquels on pioche entre 0 et la moitié de la taille totale du deck, pour simuler un avancement aléatoire dans une partie quelconque de Blackjack. Étant donné que cette étude est plus "pratique", nous nous limitons à la politique StandOn17Hard, classiquement utilisée dans les casinos. Les résultats obtenus ont été calculés pour un deck maximal d'un paquet de cartes (soit 1×52 cartes) d'une part (figure 9) et sept paquets (7×52 cartes) d'autre part (figure 10).

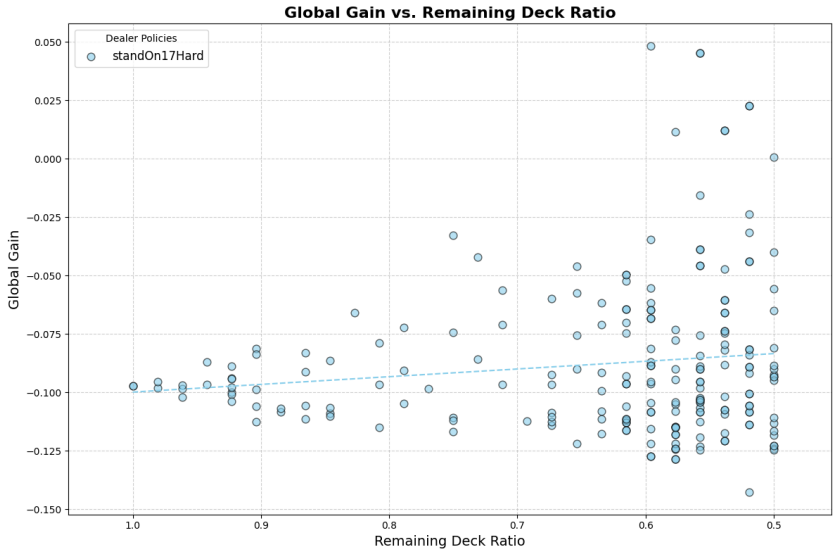

Figure 9: Gain du joueur pour des paquets aléatoires "in game" pour 1 paquet de cartes

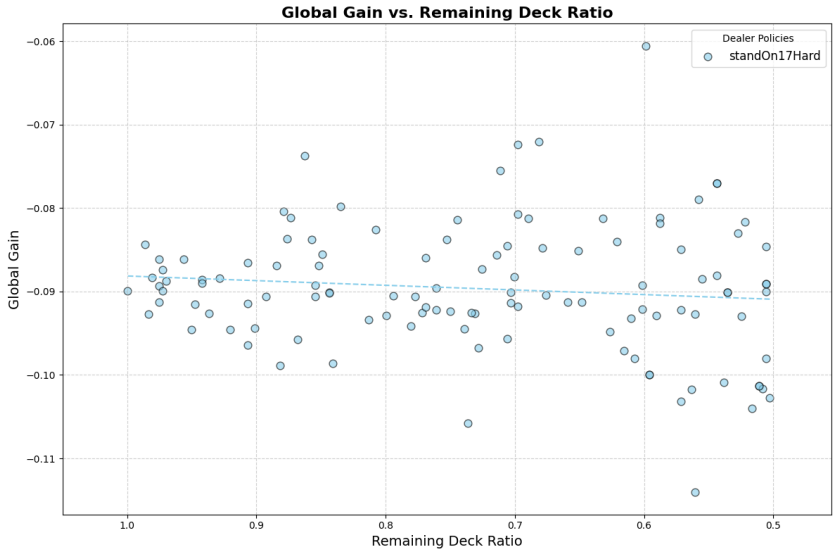

Figure 10: Gain du joueur pour des paquets aléatoires "in game" pour 7 paquets de cartes

Dans le premier graphique, on observe que plus on pioche de cartes, plus le gain moyen augmente, et que certaines parties peuvent être favorables au joueur (gain > 0). Cependant, dans le second graphe, avec 7 paquets de cartes, le gain moyen diminue au cours de l'avancée des parties. Il faudrait augmenter le nombre d'échantillons pour s'en assurer, mais l'avantage donné par le comptage des cartes semble très limité. Il faut prendre en compte le fait qu'en augmentant le nombre total de cartes, la probabilité que le paquet soit déséquilibré devient plus faible. Toutefois, il existe toujours des decks rares augmentant les chances de victoire du joueur.

Au final, le comptage des cartes donne clairement un avantage au joueur, qui s'exprime par une augmentation de son gain moyen, qui reste cependant dans la plupart des cas inférieur à 0. Cette augmentation est plus faible lorsque le nombre total de cartes augmente. Toutefois, il convient de rappeler que ces résultats ont été obtenus avec seulement les actions HIT et STAND pour le joueur. Or, il a été montré dans les questions précédentes que les autres actions parfois possibles ont tendance à augmenter le gain moyen du joueur de façon assez significative.

3 Partie III : Approche par Apprentissage par Renforcement

3.1 Modélisation par Apprentissage Supervisé

Dans cette section, nous appliquons une approche d'apprentissage supervisé pour modéliser la politique optimale du joueur en utilisant les résultats obtenus précédemment dans les calculs de politiques optimales. L'objectif est d'entraîner un modèle de réseau de neurones à reproduire la politique optimale calculée par programmation dynamique.
Les caractéristiques principales du modèle sont les suivantes :

- **Architecture du réseau de neurones** :
  - `torch.nn.Linear(27, 27, dtype=torch.float32)`
  - `torch.nn.ReLU()`
  - `torch.nn.Linear(27, 2, dtype=torch.float32)`
  - `torch.nn.Softmax(dim=1)`
- **Fonction de perte** : nous utilisons une `cross-entropy` pour évaluer la distance entre les actions prédictives du modèle et la politique optimale calculée.
- **Paramètres d'apprentissage** : l'entraînement se fait avec un batch size de 512, un nombre d'epochs de 3000, et un taux d'apprentissage de 0.001.

Cette approche permet d'obtenir une approximation de la politique optimale qui peut être déployée en temps réel, réduisant ainsi le coût de calcul associé à la programmation dynamique.

3.2 Modèle Non Supervisé

Nous avons également exploré une approche non supervisée où le modèle apprend à évaluer les performances de chaque action en s'auto-évaluant par simulation. Ce modèle suit une procédure de Monte-Carlo en réalisant plusieurs simulations de parties pour chaque action possible, et en moyennant les résultats pour ajuster ses décisions.
Les caractéristiques principales du modèle sont les suivantes :

- **Architecture du réseau de neurones** :
  - `torch.nn.Linear(27, 27, dtype=torch.float32)`
  - `torch.nn.ReLU()`
  - `torch.nn.Linear(27, 2, dtype=torch.float32)`
  - `torch.nn.Tanh(dim=1)`
- **Fonction de perte** : nous utilisons une `MSE loss` pour évaluer la distance entre les actions prédictives du modèle et les résultats obtenus par simulation.
- **Paramètres d'apprentissage** : l'entraînement se fait avec un batch size de 512, un nombre d'epochs de 3000, et un taux d'apprentissage de 0.001.

Les étapes du modèle sont :

- Pour chaque main, le modèle simule les actions `HIT` et `STAND` plusieurs fois.
- Les résultats des simulations sont utilisés pour estimer le gain attendu de chaque action.
- Les valeurs obtenues sont comparées aux prédictions a posteriori pour ajuster les décisions.

Cette approche permet une adaptation dynamique aux situations de jeu, mais elle est plus coûteuse en calcul en raison des nombreuses simulations nécessaires.

3.3 Optimisation de la Mise en Fonction de l'État du Paquet

Nous avons introduit un modèle de mise basé sur la composition actuelle du paquet, qui vise à optimiser la mise initiale du joueur. Ce modèle repose sur l'architecture suivante :

- `torch.nn.Linear(13, 13)`, suivi d'une activation `ReLU`.
- `torch.nn.Linear(13, 1)`, suivi d'une activation `ReLU`, produisant une estimation du gain attendu en fonction de l'état du paquet.

L'objectif de ce modèle est de prédire la rentabilité de chaque état du paquet afin de déterminer une mise optimale pour chaque début de partie. Les gains moyens sont estimés par des simulations multiples, afin d'obtenir une prédiction robuste pour chaque état du paquet.

3.4 Analyse des Résultats

Nous avons comparé les performances des différents modèles (supervisé, non supervisé et mise optimisée) en termes de gains moyens. Les Figures 11 et 12 montrent respectivement les résultats de performance des modèles d'apprentissage.

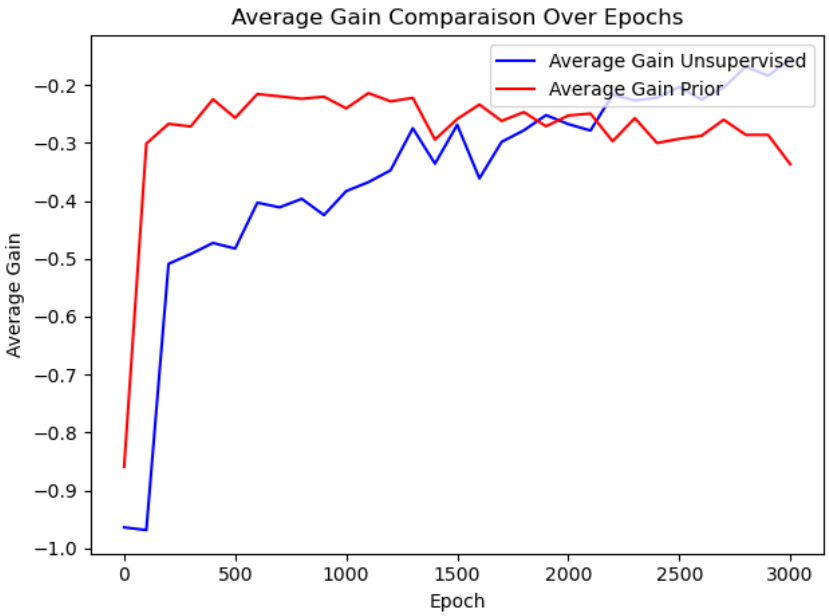

Figure 11: Comparaison des performances entre le modèle supervisé et le modèle non supervisé

Aucune des deux approches n'a réussi à surpasser la politique optimale calculée par programmation dynamique, mais elles ont permis d'obtenir des résultats satisfaisants en termes de gains moyens. Le modèle non supervisé a montré une meilleure performance que le modèle supervisé.

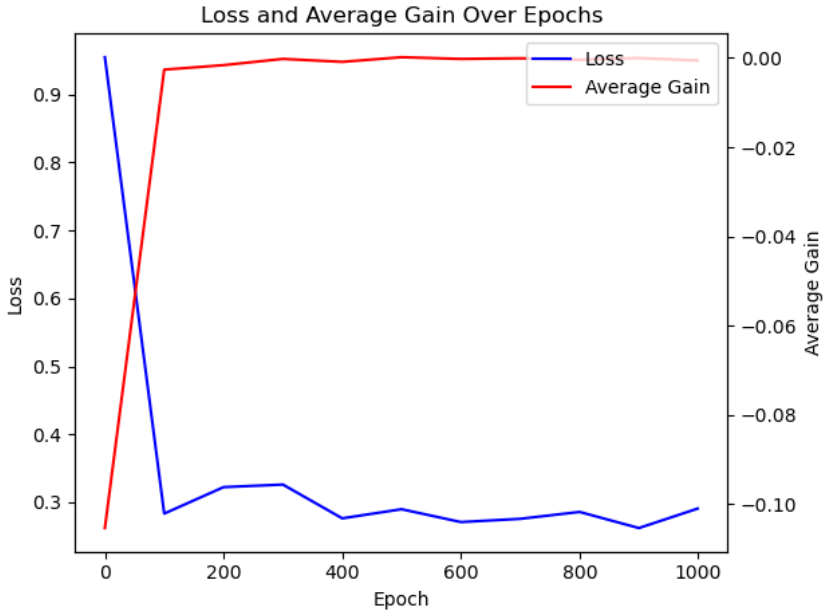

Figure 12: Résultats du modèle de mise prédictif basé sur l'état du paquet

Le modèle de mise a permis d'optimiser les gains en fonction de l'état du paquet. Toutefois, sa politique de mise est simplement de ne pas miser, car dans la plupart des cas, le gain moyen est négatif.

Conclusion

Dans ce projet, nous avons exploré des approches diverses pour calculer des stratégies optimales au Black-jack, en appliquant des méthodes de programmation dynamique et d'apprentissage par renforcement. Nos travaux ont permis d'illustrer comment les processus de décision markoviens peuvent s'appliquer à un jeu

de stratégie, en tenant compte de diverses hypothèses, notamment le tirage sans remise et la composition du paquet.
Nous avons débuté par une approche simplifiée pour calculer les politiques optimales du croupier et du joueur, et nous avons progressivement intégré des règles supplémentaires du Blackjack afin de rendre le modèle plus réaliste. L'introduction du tirage sans remise a ajouté une dimension stratégique permettant au joueur d'adapter ses décisions en fonction de l'état du paquet, bien que cela ait également augmenté la complexité de calcul.
Les approches d'apprentissage supervisé et non supervisé ont offert une alternative intéressante à la programmation dynamique, bien qu'aucune n'ait surpassé la politique optimale théorique. Le modèle non supervisé a montré une meilleure capacité d'adaptation aux situations de jeu, au prix d'un coût en calcul plus élevé. Quant au modèle de mise, il a permis d'optimiser la stratégie de pari en fonction de l'état du paquet, bien qu'il ait conclu dans la majorité des cas à une absence de mise en raison de gains moyens négatifs.

Améliorations et Perspectives

Bien que les résultats obtenus soient satisfaisants, plusieurs améliorations pourraient être envisagées :

- L'intégration de techniques de réduction de l'espace d'état pour améliorer l'efficacité des calculs en tirage sans remise.
- L'utilisation de réseaux de neurones plus sophistiqués ou de modèles d'apprentissage par renforcement avancés (comme les agents basés sur des politiques ou des méthodes d'approximation de Q-learning) pour affiner les stratégies apprises par le modèle non supervisé.

En conclusion, ce projet a mis en évidence l'utilité des processus de décision markoviens et de l'apprentissage par renforcement dans l'optimisation des décisions dans un environnement stochastique. Les perspectives offertes par ces techniques sont prometteuses, non seulement pour le Blackjack mais aussi pour d'autres applications nécessitant des stratégies adaptatives et optimisées.