PROJET D'INFORMATIQUE I

IFT 592

PROJET EN SYSTÈMES INTELLIGENTS

IFT-593

POKUS

Système d’aide au poker

Travail présenté à

M. Shengrui Wang

Par

Marc-Alexandre Côté-Harnois	07 166 997
Julien Filion	07 177 770
Simon Renaud-Deputter	07 149 640

Université de Sherbrooke

Automne 2009

Table des matières

Tables de figures :

Introduction

Le projet présenté dans ce document a été réalisé dans le cadre des cours IFT592 et IFT593 de l'Université de Sherbrooke par l'équipe Hocus, composée de Marc-Alexandre Côté, Julien Filion et Simon Renaud, à l'automne 2009.

Notre projet

Notre projet consiste en un jeu de poker distribué et un système d'aide aux usagers. Notre but est de permettre à nos joueurs d'améliorer leur jeu de poker en leur fournissant une gamme d'outils. Plusieurs outils permettent à l'usager d'obtenir plus d'informations sur le jeu comme le profil de ses adversaires et la probabilité de gagner une partie. Un autre outil conseille l'usager lorsqu'il est temps de prendre une décision. De plus, ce projet permet aux usagers de pratiquer leurs stratégies en jouant contre des adversaires artificiels.

Avant de démarrer le projet, il y avait déjà plusieurs outils déjà disponibles, mais aucun ne les réunissait tous. De plus, aucun logiciel ne permettait la suggestion d'action à l'utilisateur.

Introduction au poker

Le Poker est un jeu de cartes axé sur les probabilités et pour lequel diverses stratégies peuvent être utilisées. Le poker se joue avec des jetons et le but du jeu est d'obtenir le plus de jetons possible en obtenant la meilleure combinaison de 5 cartes ou en forçant ses adversaires à se coucher.

Il existe de nombreuses variantes de poker, par contre, pour notre projet, nous n'avons utilisé que le poker Texas Hold'Em, une des variantes les plus populaires du Poker.

Déroulement d'une partie Texas Hold'Em

Une partie Texas Hold'Em se déroule de la façon suivante :

1. On commence la partie en spécifiant quel joueur est le *Dealer*, le *Small Blind* et le *Big Blind*. Le *Small Blind* est la personne à gauche du *Dealer* et le *Big Blind* à gauche du *Small Blind*.
2. Le *Small Blind* et le *Big Blind* mettent les mises obligatoires que l'on appelle les *blinds*.
3. Le *Dealer* distribue deux cartes faces couchées à chaque joueur.
4. On démarre le tour de mise du *Preflop* (les tours de mises seront expliqués plus en détail).
5. On retourne trois cartes face ouvertes au commencement du *Flop*. Ces cartes sont partagées par tous les joueurs.
6. On fait le tour de mise du *Flop.*
7. On retourne une carte face ouverte au commencement de la *Turn*. Ces cartes sont partagées par tous les joueurs.
8. On fait le tour de mise du *Turn*.
9. On retourne une dernière carte face ouverte au commencement de la *River*. Ces cartes sont partagées par tous les joueurs.
10. On fait le tour de mise de la *River*.
11. On passe ensuite au *Showdown* où l'on montre les cartes des joueurs restants et on détermine qui a gagné les *Pots*.
12. On distribue à chaque gagnant leur partie du *Pot*.

Lors d'un tour de mise, on demande tour à tour à chaque joueur de prendre une décision. Le joueur à quatre actions possibles :

1. Lorsque la mise du joueur est égale à la mise courante, il peut parler (*Check)*. Le joueur n'augmente pas la mise et il peut continuer à jouer.
2. Si la mise du joueur est moins élevée que la mise courante, il peut égaliser la mise courante en suivant (*Call*).
3. Si la mise du joueur est moins élevée que la mise courante, il peut se coucher (*Fold*). Dans ce cas, il ne peut gagner le *pot* et il perd sa mise.
4. Finalement, le joueur peut augmenter la mise en relançant (*Raise*).

Pendant un tour de mise, s'il ne reste qu'un seul joueur qui ne s'est pas couché, il remporte le pot sans avoir à montrer ses cartes.

Un joueur est considéré *All-In* lorsqu'il met tout son argent en jeu. Ce joueur ne peut réclamer plus que sa mise totale de la part de chaque joueur.

Difficultés

Dans ce jeu, il y a plusieurs difficultés présentes. Premièrement, il s'agit d'un jeu basé sur les statistiques, il est important de bien les maîtriser pour jouer efficacement. Deuxièmement, il y a beaucoup d'informations cachées. Par exemple, on ne connait pas les cartes des autres joueurs. Troisièmement, il est difficile de déterminer l'intention d'un adversaire, car il existe une multitude de stratégies différentes et nous possédons généralement peu d'information sur le jeu de l'adversaire. De plus, un joueur peut décider de changer de stratégie à tout moment, ce qui est très difficile à détecter. Finalement, les joueurs vont généralement prendre des décisions imprévisibles, par exemple il est possible qu'un joueur prenne deux décisions différentes face à une même situation.

En raison de ces difficultés, un bon joueur doit avoir une bonne gestion des risques en évaluant les montants que l'on peut remporter et les montants dépensés. Il doit aussi avoir une bonne connaissance des probabilités pour évaluer ses chances de remporter les mises. De plus, le joueur doit pouvoir analyser ses adversaires pour mieux les déjouer.

Plate-forme de poker

La plate-forme de poker est composée d'un serveur de poker et d'une application cliente. Cette partie du projet était en grande partie de la conception logicielle. Le protocole de communication entre le client et le serveur est annexé à la fin du document.

Serveur de poker

Le serveur de poker est une application TCP/IP qui permet au client de jouer au poker. Il est composé d'un serveur de lobby (hall d'entrée), de gestionnaires de tables et de simulateurs de parties.

Simulateur de partie (*HoldemTable*)

Le simulateur de partie est la base de la plate-forme de poker. Il simule toutes les étapes d'une partie de poker en respectant les règles du jeu.

Le simulateur de partie supporte trois variantes du poker Texas Hold'Em, soit le *No Limit*, le *Pot Limit* et le *Fixed Limit*. Ces trois variantes sont très semblables et se différencient principalement par le montant maximum qu'il est possible de miser à chaque décision. Dans la variante *No Limit* il est possible de miser tous ces avoirs. Dans la variante *Pot Limit*, on ne peut augmenter la mise du montant total sur la table. Dans la variante *Fixed Limit*, on ne peut miser que d'un montant prédéfini.

Afin de pouvoir comparer la force des mains rapidement et efficacement, nous avons utilisé le code source de l'application *Steve Brecher Hand Eval*.

Le simulateur de partie est construit pour travailler sur l'interface *IPlayer* de sorte qu'il ne connaisse pas le fonctionnement des joueurs. Ainsi, il est facile d'ajouter des joueurs de différents types au simulateur. On peut donc utiliser des joueurs artificiels, des joueurs locaux et des joueurs réseau sur la même table. On peut noter que notre système utilise principalement le joueur *MessageNetworkPlayer* qui permet de communiquer avec un client en utilisant le protocole défini en annexe.

Serveur de lobby (*ServerLobby*)

Le serveur de lobby est le point d'entrée des clients, il attend des connexions en écoutant sur un port explicite. Le serveur du lobby possède son propre port qui doit être connu du client. Pour utiliser notre système, un client doit tout d'abord se connecter au serveur de lobby. Une fois authentifié, il peut décider de créer une nouvelle table, si le nombre[1] de ports libres le permet. Un client peut également décider de voir la liste des tables présentes sur ce serveur. En plus des informations sur la table, cette liste contient également le numéro de port du gestionnaire de table, ce qui permet au client de joindre la table.

[1] Modifiable via le fichier de configuration. (*config.properties*)

Gestionnaire de table (*TableManager*)

La tâche du gestionnaire de table est de gérer les connexions pour une table bien précise. Lorsque le serveur de poker crée une table, on crée aussi un gestionnaire de table qui s'occupe d'écouter sur le port qui est associé à la table. Lorsqu'un client désire se connecter à une table, on l'ajoute à la table (*HoldemTable*) qui s'occupera dorénavant de gérer la communication avec le joueur.

Client

La partie client est une application TCP/IP qui permet aux usagers d'utiliser le jeu de poker à distance. La partie client a été conçue de façon à permettre l'internationalisation, c'est-à-dire que tous les littéraux sont gardés dans plusieurs[2] fichiers externes au programme.

Voici le diagramme de classe résumant l'architecture côté client :

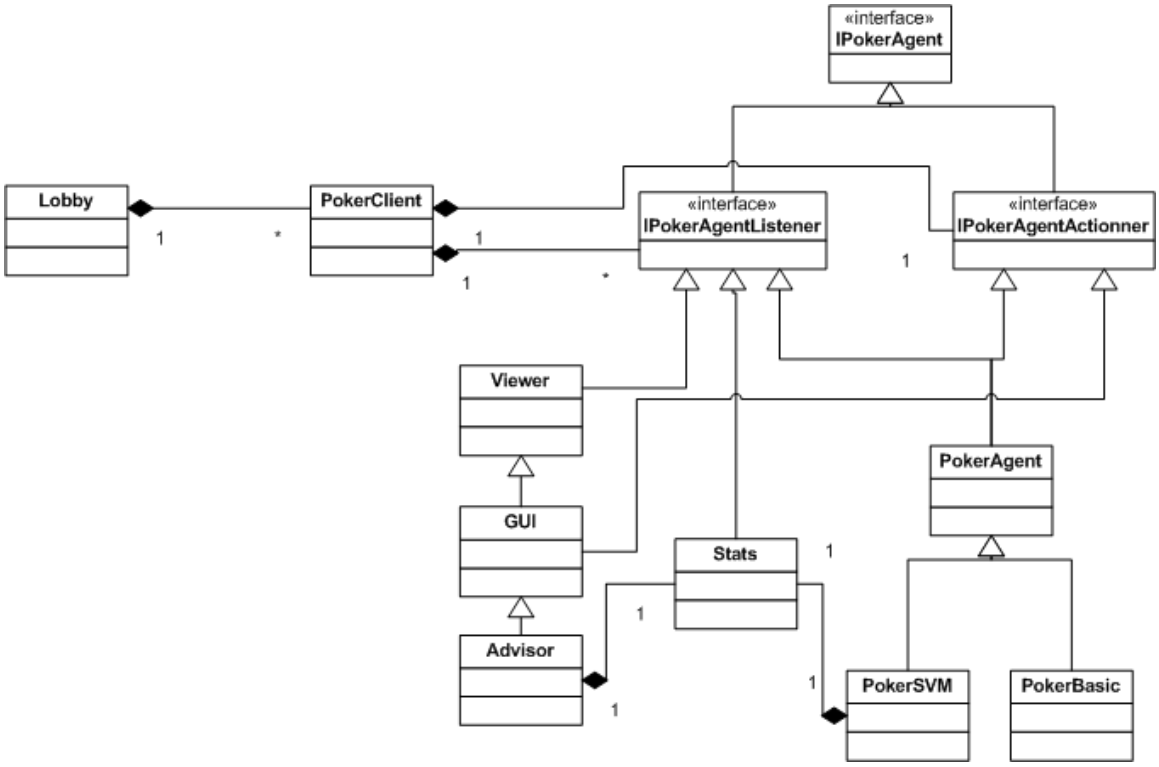


Figure 1 – Architecture côté client

[2] Un fichier pour chaque langue.

Lobby

Le lobby est la page d'accueil du côté client. Avant de pouvoir commencer à jouer, l'utilisateur doit se connecter à un serveur de lobby (*ServerLobby*). Dans cette fenêtre, l'usager peut voir, une fois connecté, la liste des tables disponibles sur le serveur. C'est aussi à partir de cet endroit qu'il est possible d'ajouter une nouvelle table et de rejoindre une partie en cours. Un lobby ne peut gérer qu'un seul joueur à la fois.

Figure 2 – Lobby

Fenêtre de connexion

À partir de cette fenêtre, l'utilisateur peut décider de se connecter en tant qu'humain ou bien en tant qu'agent intelligent. Indépendamment du choix, l'usager doit fournir un nom de joueur[3]. S'il a choisi l'option d'être un agent, alors il doit fournir le type d'agent intelligent et indiqué si un visionneur doit être attaché à ce joueur. Par la suite, il ne reste qu'à entrer les informations du serveur : adresse (IP ou le nom de domaine) et le numéro de port.

Ensuite, lorsque l'usager appuie sur le bouton de confirmation, un message est envoyé au serveur afin d'authentifier le client du joueur. De plus, un second message est envoyé automatiquement par le client afin de rafraîchir la liste des tables disponibles sur le serveur.

Figure 3 – Fenêtre de connexion

[3] Le nom d'un joueur ne peut être présent qu'une seule fois sur une table.

Fenêtre de création de tables

À l'aide de cette fenêtre, un utilisateur a la possibilité de créer une nouvelle table et d'en spécifier les caractéristiques. Il peut ainsi déterminer le nom de la table, le nombre de joueurs maximum qu'elle accueillera, la mise initiale des parties et la variante de poker *Texas Hold'em* parmi les trois proposées.

Ensuite, lorsque l'usager appuie sur le bouton de confirmation, un message st envoyé au serveur afin de créer une nouvelle table. De plus, le joueur est automatiquement ajouté à cette table.

Une réponse du serveur indiquera le numéro de port de la nouvelle table (création réussie) ou bien indiquera un échec (-1). Par la suite, un second message est envoyé automatiquement par le client afin de joindre la nouvelle table. Finalement, un dernier message est envoyé lui aussi automatiquement afin de rafraîchir la liste de tables disponibles sur le serveur.

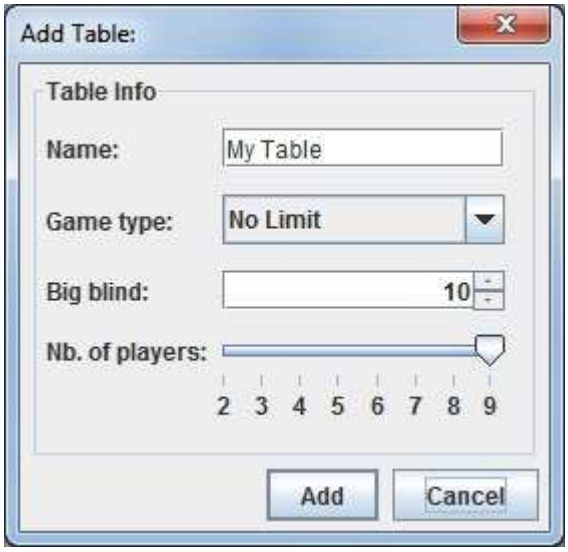


Figure 4 – Création de table

Joindre une table

Afin de rejoindre une partie en cours, l'utilisateur n'a qu'à sélectionner la table désirée et à cliquer sur le bouton servant à joindre une table. À ce moment, un message est envoyé au bon gestionnaire de table (*TableManager*) en se basant sur le numéro de port associé à chaque table. La réponse du serveur indiquera le numéro de siège du joueur à cette table (réussie) ou bien -1, si échec (par exemple, si un usager est déjà assis à une certaine table, une nouvelle tentative pour joindre la table résultera par un échec)

Options avancées

Ces options permettent à l'usager d'ajouter et de retirer des observateurs sur leur joueur. Il en existe différentes sortes. Le visionneur en est un, ainsi que le collecteur de statistiques. L'architecture du côté client permet de créer et d'ajouter facilement de nouveaux observateurs. (Voir diagramme de classes)

Par exemple, on démarre une partie en tant qu'agent, mais on a oublié de lui associer un visionneur. Or, à partir des options avancées, l'usager peut ajouter un visionneur dynamiquement sur son joueur.

Figure 5 – Options Avancées

PokerClient

La classe PokerClient s'occupe de communiquer avec le serveur (*HoldemTable*). Elle s'occupe donc d'extraire les informations des messages reçus. Elle possède également sa propre représentation d'une table de poker, qu'elle met à jour à l'aide des informations envoyées par le serveur. Lorsqu'un usager joint une table, le serveur commence par lui envoyer un message TABLE_INFOS[4]. Avec ce message, il est possible de reconstruire une table de poker avec toutes ses informations. Par la suite, pour chaque nouvel événement, la table sera modifiée, dans un premier temps, afin de refléter les changements et dans un deuxième temps, PokerClient s'occupera d'avertir chaque observateur leur indiquant que la table a été mise à jour à la suite d'un événement.

[4] Voir la section sur les messages utilisés par le protocole de communication.

Fenêtre de jeu

Le client de poker est une fenêtre représentant la partie en cours pour une table donnée. Il existe trois types de fenêtre :

1. Viewer : le visionneur permet d'observer une partie selon la vision d'un joueur, donc aucune interaction n'est possible.
2. GUI : le GUI offre la même vision que le Viewer, mais en plus cette fenêtre permet à l'utilisateur de jouer au poker contre d'autres joueurs et donc offre des boutons pour interagir avec la table de poker.
3. Advisor : l'Advisor est un sous-type du GUI, sauf qu'en plus de permettre l'interaction avec la table, cette fenêtre offre des outils et des conseils à l'usager. (Statistiques des adversaires, probabilité de gagner, profil d'un adversaire et le prochain coup optimal à jouer)

Viewer

Avec ce type de fenêtres, un usager peut associer autant de visionneurs qu'il le désire à son joueur. Une fois un nouveau visionneur ajouté à la liste des observateurs, il a automatiquement accès à la représentation de la table contenue dans le PokerClient, il peut ainsi construire la représentation graphique de la partie en cours.

GUI

La classe GUI dérive de la classe Viewer, cela permet d'afficher la représentation de la partie à l'écran. Par contre, en plus d'être un observateur, cette classe implémente l'interface *IPokerActionner*, ce qui rend possible l'interaction de l'usager avec la table. Lorsque le serveur envoie un message TAKE_ACTION[5], le GUI rend disponibles les boutons associés aux actions permises par le serveur. Par la suite, le joueur peut prendre sa décision en se basant sur les informations affichées à l'écran.

[5] Voir la section sur les messages utilisés par le protocole de communication.

Advisor

La classe Advisor est une sous-classe de GUI et par conséquent permet un affichage graphique de la partie, en plus de l'activation des boutons interagissant avec la table. En plus de cela, elle offre quatre outils importants servant à aider l'utilisateur :

- Statistiques : sommaire des informations recueillies sur les adversaires et présentées sous forme de statistiques.
- Profil des adversaires : Profile type des adversaires reflétant leur agressivité et leur assurance.
- Probabilités de gagner : chance que possède le joueur de remporter la partie.
- Suggestion : choix proposé par le système d'aide quant à la prochaine action à prendre. Ce choix est la décision que prendrait un joueur artificiel (celui à base d'un SVM).

L'approche et les algorithmes utilisés pour fournir ces informations à l'usager seront discutés en détail dans la prochaine section (Système d'aide au poker).

Système d'aide au poker

Le système offre plusieurs outils aux usagers pour améliorer leur jeu. Il utilise la plateforme de poker décrite précédemment et plus particulièrement l'Advisor. Voici l'interface la plus importante du système d'aide :

Figure 6 – Interface du système d'aide

Règles de poker (charte)

Dans le menu d'aide du lobby, on peut y retrouver une charte affichant toutes les mains possibles au poker. Elles sont classées en ordre de force de la plus faible (bas) à la plus forte (haut). Cette fenêtre peut être laissée ouverte tout au long d'une partie permettant ainsi à l'usager de s'y référer en cas de doute.

Figure 7 – Charte des forces des mains au poker

Probabilité de gagner

Cette information permet à l'usager de savoir quelles sont ses chances de remporter la mise étant donné le nombre d'adversaires, les cartes qu'il a en main et les cartes visibles sur la table. Cette probabilité est calculée en utilisant la méthode de Monte Carlo qui consiste à répéter une expérience un grand nombre de fois afin d'en tirer une moyenne qui reflète la réalité. Dans le cadre du système d'aide, une expérience consiste à simuler la fin de la partie, en considérant les données de départ. Pour ce faire, une carte est tirée aléatoirement pour chaque carte inconnue du joueur. Ensuite, la probabilité représente le nombre de fois que le joueur a gagné sur le nombre de simulations (un million dans notre cas).

En plus de calculer la probabilité de gagner, on calcule l'écart type de la probabilité de gagner sur la prochaine carte tirée. Cette information n'est pas disponible au premier tour de mise (*Preflop*), on doit donc tirer trois cartes avant le prochain tour. Lors du tour de mise de la *River*, l'écart type est de 0, car aucune carte n'est tirée par la suite.

Statistiques

Un autre outil accessible à l'usager est la table regroupant des statistiques sur les adversaires assis à la même table que l'utilisateur. Les statistiques représentent des situations les plus courantes au poker.

Il y a principalement deux façons d'obtenir de l'information sur un joueur. On peut lire les signes non verbaux et l'évaluer selon certaines statistiques. Comme nous avons fait une approche logicielle, nous avons fait la cueillette d'information avec seulement les statistiques. Il y a une grande quantité de statistiques que nous pouvons recueillir sur les joueurs.

Par exemple, il y a une statistique qui calcule le nombre de fois qu'une personne a été le dernier joueur à avoir misé lors du *Preflop*. Pour être plus précis, en plus de garder le nombre de fois où quelqu'un a été le dernier à miser, on peut conserver sa position et la phase de la partie (*Preflop*, *Flop*, *Turn* ou *River*). Toutefois, plus on veut d'information, plus on doit avoir de données. Donc, nous n'avons gardé que les statistiques les plus discriminantes.

En plus d'être pratique pour les usagers, on utilise les statistiques pour entrainer les Support Vector Machine et pour classifier les adversaires selon différents profils.

Voici le tableau de statistiques visible par le joueur principal :

Figure 8 – Tableau de statistiques des adversaires courants.

Classification

Le système d'aide offre la possibilité à l'utilisateur de voir le profil de ses adversaires. La méthode utilisée par le système afin de déterminer la classe d'un joueur est celle à base de règles et sera détaillée dans la prochaine section. Pour voir le profil d'un adversaire, il suffit de placer sa souris au-dessus du joueur désiré et un message info-bulle affichera les informations concernées.

Suggestion

Un des meilleurs outils du système d'aide est le fait de suggérer la prochaine décision à prendre. Pour faire afficher cette suggestion, il suffit d'activer les conseils à partir du bouton prévu à cet effet. Ensuite, un message sera affiché au-dessus des boutons d'actions. L'action suggérée est celle qu'un joueur artificiel basé sur les SVMs prendrait. En effet, lorsque l'on rejoint une table, un joueur artificiel est automatiquement ajouté aux observateurs. Donc, lorsque le serveur envoie un message indiquant au client que c'est à son tour, le système d'aide demande au joueur intelligent son action et l'affiche en tant que conseil. Le fonctionnement interne du joueur artificiel basé sur les SVMs sera couvert dans une prochaine section.

Classification

Puisque chaque joueur possède une stratégie unique, il est difficile de prévoir leur prochaine action qui varie considérablement en fonction de leur comportement. C'est pourquoi les joueurs de poker ont tendance à regrouper leurs adversaires en différentes catégories. Cette astuce permet de développer des stratégies efficaces contre un groupe de personnes. Par contre, il est très difficile de classifier un joueur dû à plusieurs raisons. Par exemple, les décisions des joueurs sont non déterministes. Or, dans la littérature en général, ils classifient souvent les joueurs selon trois critères : leur niveau d'assurance (prudent/fonceur), leur degré d'agressivité (passif/agressif) avant le *Flop* et leur degré d'agressivité après le *Flop*.

Basée sur des règles

Pour y parvenir, nous avons utilisé des règles définies par un expert du domaine en utilisant les statistiques recueillies. Un joueur prudent est un joueur qui n'entre pas souvent lors du *Preflop*, car il joue seulement avec des mains qui sont environ dans les 10 % meilleures combinaisons de deux cartes possibles au *Preflop*. D'un autre côté, le joueur fonceur va entrer plus souvent. Pour ce qui est du degré d'agressivité, un joueur agressif va miser de gros montants et va le faire souvent alors qu'un joueur passif va miser de petites sommes et moins souvent.

Afin de déterminer si un joueur est prudent/fonceur, on regarde la statistique calculant le nombre de fois qu'il a mis de l'argent dans le pot volontairement. Pour ce qui est du niveau d'assurance, on prend en considération plusieurs statistiques qui sont détaillées sur le site de Holdem Manager[6].

[6] http://www.holdemmanager.net/faq/afmviewfaq.aspx?faqid=155

Modèle supervisé

Une autre façon de procéder pour déterminer la classe d'un joueur c'est d'utiliser une quantité importante d'historiques de parties afin d'entraîner un classificateur à partir d'informations qu'on y extrait. Par contre, ce type de modèle nécessite de connaître la classe réelle du joueur au moment de l'entraînement. Or, comme nous n'avions pas cette information, nous avons laissé tomber cette approche. Voici un diagramme montrant comment nous aurions pu procéder :

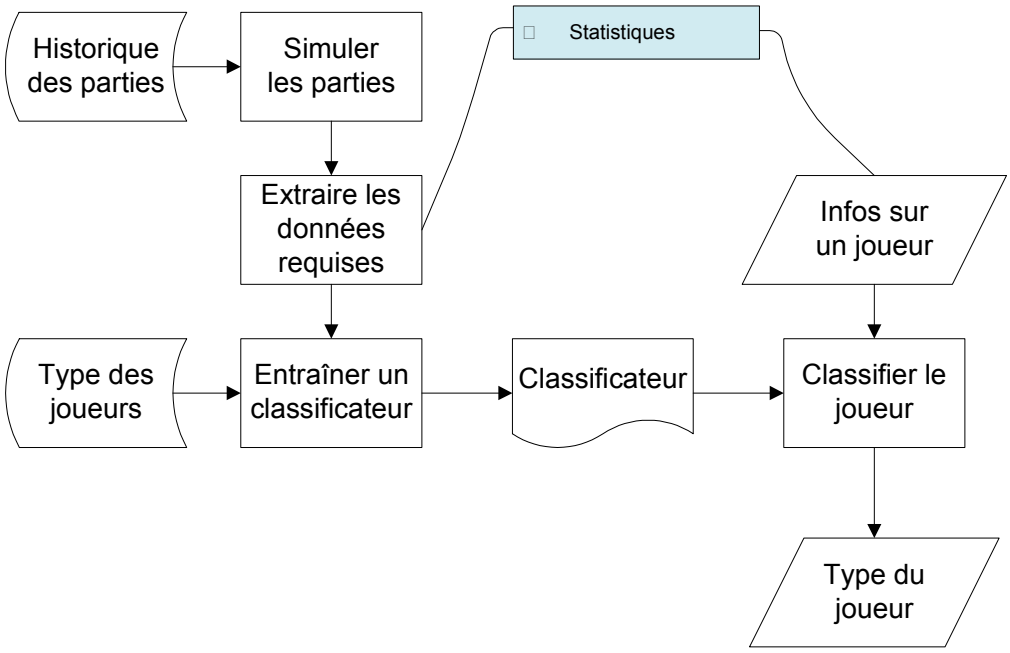


Figure 9 – Fonctionnement d'une classification supervisé

Modèle non supervisé

Une autre approche, semblable au modèle supervisé, est d'utiliser les historiques pour y en extraire les informations désirées et ensuite les segmenter. Contrairement à la classification, lors de l'entraînement aucune information sur la classe réelle du joueur n'est requise. Par contre, l'inconvénient majeur est que les groupes résultants de la segmentation ne sont pas intuitifs pour les utilisateurs. À moins d'étiqueter chaque segment, cette méthode ne peut être utilisée pour notre projet puisque le but est de créer un système d'aide et doit donc être en mesure de fournir des informations utiles aux usagers. Voici tout de même son fonctionnement :

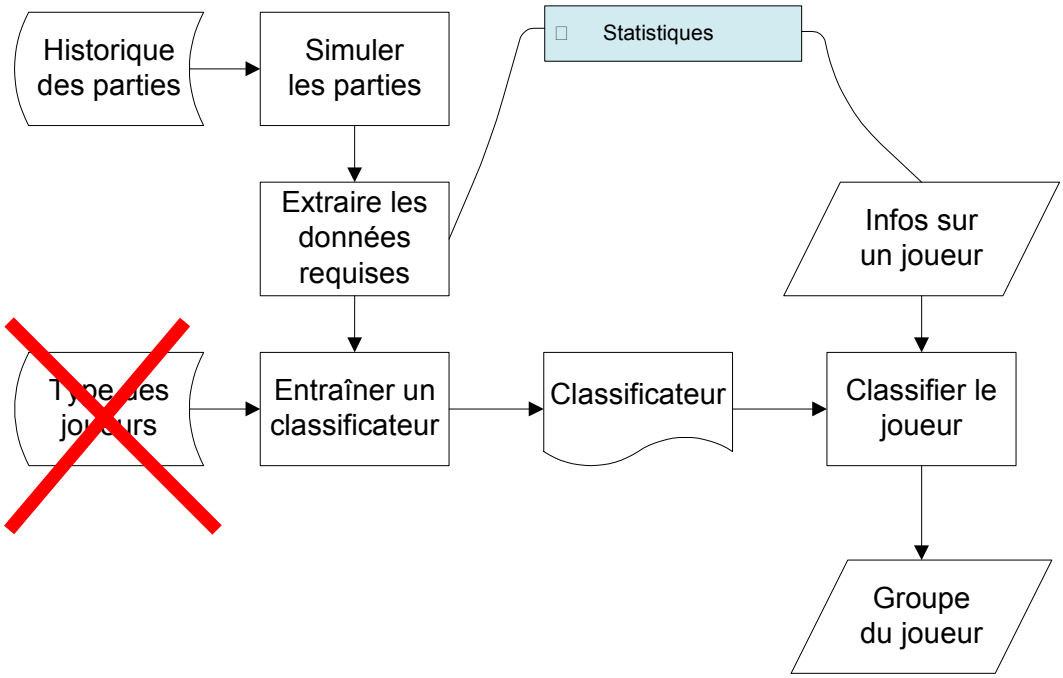


Figure 10 – Fonctionnement de la classification non supervisé

Joueur intelligent

Une partie importante du projet a été la création de joueurs artificiels.

Un des buts recherchés dans la création de joueur artificiel est d'offrir une simulation réaliste aux usagers de notre système. En ajoutant des joueurs artificiels sur une table, un joueur a la possibilité de jouer seul afin de développer ses stratégies. De plus, il est possible d'ajouter des joueurs de différents calibres ce qui permet au débutant d'apprendre progressivement et aux experts d'avoir plus de défi.

Un autre objectif que nous avions était de proposer des actions à l'usager. Cette option permet au débutant de valider leurs actions et aux joueurs plus expérimentés de comparer leurs stratégies.

Par contre, réaliser des joueurs artificiels réalistes et doués est une tâche particulièrement complexe. En effet, il faut prendre en considération qu'un joueur doué doit tenir compte du type de ses adversaires, des probabilités, des montants à risquer et il doit faire attention à ne pas dévoiler ses tactiques.

Aussi, nous avons remarqué que les joueurs humains ont tendance à modifier leur comportement en prenant beaucoup plus de risque lorsqu'ils jouent avec du faux argent, alors que des joueurs artificiels restent constants.

Basé sur des règles

Il est possible de faire un joueur qui se base sur des règles préfixées pour prendre une décision. Utiliser cette technique demande par contre une bonne connaissance du domaine. Par contre, puisque cette solution est simple à implanter, nous avons décidé de l'utiliser comme premier essai. Nous avons utilisé les conseils de notre expert pour générer un système de règle qui puisse jouer de manière correcte.

Ce système souffre de certaines lacunes, par exemple le joueur ne tentera pas de bluffer et il n'évalue pas le jeu de ses adversaires.

Dans nos règles, nous utilisons la probabilité de gagner ainsi que deux seuils calculés à partir du nombre de joueurs sur la table.

Prob = probabilité de gagner la partie
x = Nombre de joueurs en jeu.
Seuil1 = 0.02 + √(3/(5*(x-0.75)))
Seuil2 = -0.38 + √(4/(x+3))

Règle :

Prob >= seuil1 :
    Le joueur augmente la mise (raise).
Prob < seuil1 et Prob >= seuil2 :
    Le joueur Check si il le peut ,sinon il Call.
Prob < seuil2
    Le joueur Check si il le peut, sinon il se couche (Fold).

Algorithme génétique :

Les paramètres utilisés dans notre système à base de règle ont été fixés de manière empirique. Nous avons donc décidé d'optimiser ce joueur en utilisant un algorithme génétique. L'algorithme génétique nous générait plusieurs joueurs artificiels et nous les faisions jouer les uns contre les autres pour ne conserver que les meilleurs. Malheureusement, la simulation des parties prenait trop de temps, ce qui a limité le nombre de joueurs testé.

Support Vector Machine

Comme deuxième approche, nous avons décidé de générer un joueur artificiel qui utilise des Support Vector Machine (SVM). Notre but était de faire un SVM qui est capable de prendre une décision appropriée à chaque situation. Voir l'annexe 1 pour un diagramme montrant le processus relié au SVM.

Afin de générer un SVM qui prend de bonnes décisions, nous avons décidé de l'entrainer pour qu'il copie les décisions d'un joueur gagnant. Pour ce faire, nous avons utilisé environ 800 000 parties jouées sur internet par le même joueur. Ces historiques de parties contiennent toute l'information des parties visibles à notre joueur. Par contre, notre joueur utilisait beaucoup les statistiques de ces adversaires qu'il compilait au fur et à mesure de ses parties et nous n'avions pas ces statistiques.

À l'aide de ces données, nous avons simulé chaque partie une après l'autre en recalculant les statistiques. Pour chaque décision que notre joueur a prise, nous avons généré un vecteur qui représentait l'environnement pour la prise de décision. Cet environnement est composé des cartes en jeu, des jetons en jeu, des actions prisent depuis le début de la partie et des statistiques de chaque adversaire. Nous avons associé à chaque vecteur une classe qui représente l'action du joueur (Check, Call, Fold, Raise). Comme le SVM donne de meilleurs résultats si toutes les dimensions du vecteur ont le même intervalle de valeurs possibles, nous avons normalisé toutes ces valeurs entre 0 et 1. Grâce à ces vecteurs, nous avons entrainé et testé différents SVM qui copiaient le comportement de notre joueur gagnant.

Pour entrainer notre SVM, nous avons utilisé l'outil LIBSVM[7] car on pouvait l'utiliser facilement à partir d'une application Java. Par contre, nous avons éprouvé un problème avec cet outil, car il ne pouvait classifier que de manière binaire. Pour corriger la situation, nous avons fait deux versions de nos vecteurs. La première classait une décision comme « se coucher » ou « continuer à jouer » et la deuxième était « augmenter la mise » ou « ne pas augmenter la mise ». De plus, le processus de prise de décisions au *Préflop* est assez différent du *Postflop*, c'est pourquoi nous avons décidé de séparer tous nos vecteurs en deux catégories, soit *Préflop* et *Postflop*. Aussi, nous avons conservé une petite partie de nos données pour permettre de tester les modèles générés. Ce qui nous donne finalement huit ensembles de données distinctes pour représenter trois séparations : *Fold/Raise*, *Preflop/Postflop*, entrainement/test.

Par la suite, nous avons entrainé les quatre SVM nécessaires pour copier les actions de notre joueur gagnant. Pour l'entrainement les SVM, nous n'avons pas eu la chance de bien essayer tous les paramètres de l'algorithme, nous sommes donc conscients que les résultats obtenus peuvent encore être améliorés.

[7] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines, 2001. Logiciel disponible sur http://www.csie.ntu.edu.tw/~cjlin/libsvm

Finalement, pour prendre une décision en utilisant nos SVM, on doit tout d'abord générer un vecteur qui représente l'environnement dans lequel la décision doit être prise. Avec ce vecteur, on peut interroger nos quatre SVM pour obtenir l'action à prendre. Voici comment on procède pour prendre une décision :

1. Tout d'abord, si on se trouve au *Preflop*, on utilise les deux SVM du *Preflop* sinon on prend ceux du *Postflop*.
2. Par la suite, on interroge le SVM *Fold* pour savoir si on doit se coucher.
3. Si on ne se couche pas, on interroge le SVM *Raise* pour savoir si on doit augmenter la mise.
4. Finalement, si nous n'augmentons pas la mise, on *Check* si c'est possible, sinon on *Call*.

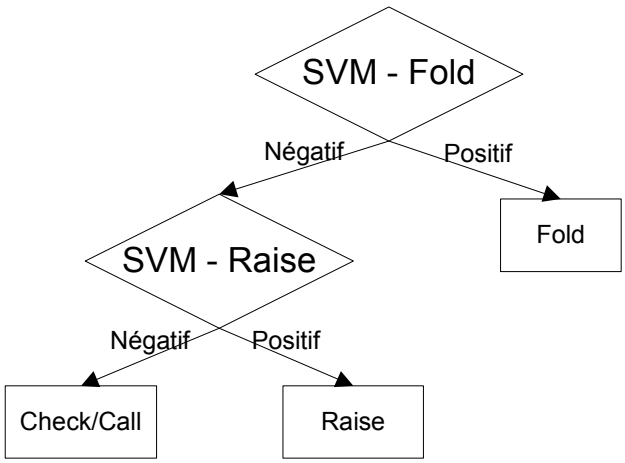


Figure 11 – Prise de décisions avec SVM

Résultats

Grâce à nos ensembles de tests, nous avons pu valider la validité de nos résultats. Chaque SVM a été testé avec un ensemble de 10 000 vecteurs qui n'étaient pas inclus dans les données d'entrainement.

Se coucher au *Preflop* (preflopFold.svm) :

Accuracy :	88.89%
Precision :	93.01%
Recall :	94.38%

Augmenter la mise au *Preflop* (preflopRaise.svm) :

Accuracy :	88.94%
Precision :	92.55%
Recall :	91.71%

Se coucher au *Postflop* (postflopFold.svm) :

Accuracy :	94.11%
Precision :	81.23%
Recall :	89.52%

Augmenter la mise au *Postflop* (postflopRaise.svm) :

Accuracy :	74.35%
Precision :	73.03%
Recall :	51.67%

Glossaire :

All-In :

Ce dit d'un joueur qui à miser tous les jetons de sa banque.

Big blind :

Le big blind est une mise obligatoire qui doit être posée sur la table par un joueur au début d'une partie. La valeur du big blind est le double de celle du small blind. Le joueur qui pose le big blind est aussi appelé big blind.

Blind :

Le blind est une mise obligatoire qui doit être placée au début d'une partie. Au poker Texas Hold'Em, deux blinds doivent être placées au début de chaque partie, soit le big blind et le small blind.

Banque :

La banque d'un joueur correspond aux jetons qu'il possède dans une partie de poker. Les jetons qu'il a misés dans une partie en cours ne sont pas inclus dans la valeur de sa banque.

Call (suivre, égaliser) :

Décision que peut prendre un joueur lorsque c'est à son tour de parler. Si un joueur prend cette action, il doit ajouter le montant nécessaire à sa mise pour égaliser la mise courante.

Check (parler) :

Décision que peut prendre un joueur lorsque c'est à son tour de parler. Lorsque la mise du joueur est égale à la mise courante, il peut checker. Il n'augmente pas la mise et peut continuer à jouer.

Dealer (donneur) :

Le dealer est le joueur qui distribue les cartes.

Early position :

Nom donné aux sièges situés tout de suite après le *Big Blind.* (Varie en fonction du nombre de joueurs à la table.)

Flop :

Au poker Texas Hold'Em, c'est la période où l'on tire les trois premières cartes face ouverte sur la table. On y joue aussi le deuxième tour de mise.

Fold (se coucher) :

Décision que peut prendre un joueur lorsque c'est à son tour de parler. Si un joueur se couche, il n'a pas à mettre plus de jetons sur la table, mais il ne peut plus gagner les pots de cette partie.

Hole (carte) :

Les cartes Hole sont les deux cartes que possède un joueur au Texas Hold'Em.

Late position :

Nom donné aux sièges situés tout de suite avant le *Big Blind.* (Varie en fonction du nombre de joueurs à la table.)

Main :

Au poker Texas Hold'Em, c'est la meilleure combinaison de cinq cartes parmi les sept visibles que peut faire un joueur.

Middle position :

Nom donné aux sièges situés entre les Early positions et les Late positions. (Varie en fonction du nombre de joueurs à la table.)

Plateau (board) :

Nom donné à l'endroit où sont déposés les jetons et les cartes d'une partie de poker.

Postflop :

Au poker Texas Hold'Em, il s'agit de la période qui suit le flop. Elle inclut les tours de mise du flop, du turn et de la river.

Pot :

Les jetons qui se trouvent au milieu de la table et qui peuvent être gagnés par les joueurs.

Pot secondaire :

Un pot secondaire est créé lorsqu'un joueur se met all-in, car ce joueur ne peut remporter plus que sa mise de la part de chaque joueur.

Preflop :

Au poker Texas Hold'Em, il s'agit de la période qui précède le flop. Elle inclut la mise des blinds et le tour de mise du preflop.

Raise (relancer) :

Décision que peut prendre un joueur lorsque c'est à son tour de parler. Le joueur augmente la valeur de la mise courante.

River :

Au poker Texas Hold'Em, c'est la période où l'on tire la dernière carte face ouverte sur la table. On y joue aussi le dernier tour de mise.

ShortStack :

Au poker Texas Hold'Em, stratégie qui consiste à jouer avec un petit nombre de jetons.

Showdown :

Au poker Texas Hold'Em, c'est la période suivant le dernier tour de mise. On y montre les cartes des joueurs restants, on détermine les gagnants et on distribue les pots aux gagnants.

Small blind :

Le small blind est une mise obligatoire qui doit être posée sur la table par un joueur au début d'une partie. La valeur du small blind est la moitié de celle du big blind. Le joueur qui pose le small blind est aussi appelé small blind.

SVM :

Acronyme pour Support Vector Machine.

Table :

Endroit où des joueurs s'assoient pour y jouer une ou plusieurs parties de poker**.**

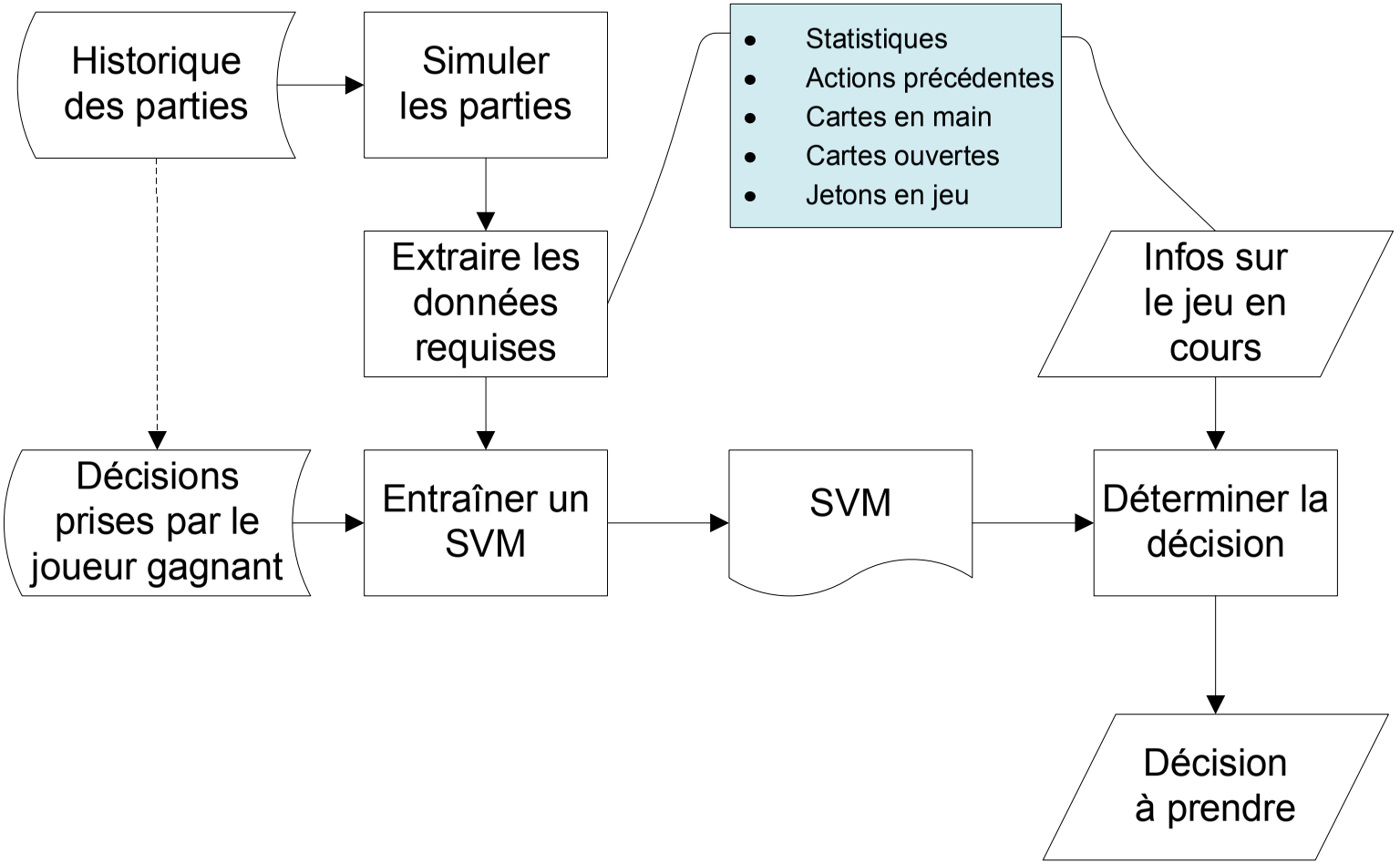
Turn :

Au poker Texas Hold'Em, c'est la période où l'on tire la quatrième carte face ouverte sur la table. On y joue aussi le troisième tour de mise.

Under the gun :

Premier joueur à parler lors d'un tour de mise.

Annexe 1 : Diagramme Support Vector Machine



Annexe 2 : Protocole de communication

Communication de la table au client

État de la table :

Envoyé lorsqu'un joueur joint une nouvelle table. Ce message contient toutes les informations sur la partie en cours.

Syntaxe := TABLE_INFOS;*totalPotAmount*;*nbPot*; *potsAmount boardCards nbSeat*; *seatsInfos*

potsAmount :=*potAmount;* (répéter 9 fois)

boardCards :=*boardCard;* (répéter 5 fois)

seatsInfos := *seatInfos* (répéter 9 fois)

seatInfos := *emptySeat* |
playerInfos

emptySeat := *noSeat*;false;

playerInfos := *noSeat*;true;*playerName*;*moneyAmount*;*card1*;*card2*;*isDealer*; *isSmallBlind*;*isBigBlind*;*isPlaying*;*timeRemaining*;*betAmount;*

Paramètre	Type	Définition
totalPotAmount	int	Montant total sur la table.
nbPot	int	Nombre de *Pots*.
potAmount	int	Nombre de jetons de chaque *Pot*.
boardCard	int	Identificateur d'une carte du plateau.
nbSeat	int	Nombre de sièges de la table.
noSeat	int	Numéro de siège d'un joueur.
playerName	String	Nom du joueur.
moneyAmount	int	Le nombre de jetons dans la banque du joueur.
card1	int	Identificateur de la première carte du joueur.
card2	int	Identificateur de la deuxième carte du joueur.
isDealer	bool	Vrai si le joueur est le *Dealer.*
isSmallBlind	bool	Vrai si le joueur est le *Small Blind*.
isBigBlind	bool	Vrai si le joueur est le *Big Blind*.
isPlaying	bool	Vrai si le joueur joue dans la partie en cours.
timeRemaining	int	Temps restant au joueur pour prendre sa décision.
betAmount	int	Montant de la mise du joueur.

Début de partie :
Envoyé lorsque l'on démarre une nouvelle partie.
Syntaxe := GAME_STARTED;*noSeatDealer*;*noSeatSmallBlind*;*noSeatBigBlind*;

Paramètre	Type	Définition
noSeatDealer	int	Numéro de siège du *Dealer*.
noSeatSmallBlind	int	Numéro de siège du *Small Blind*.
noSeatBigBlind	int	Numéro de siège du *Big Blind*.

Début d'une action:
Envoyé lorsqu'un joueur doit prendre une décision.
Syntaxe := PLAYER_TURN_BEGAN;*noSeat*;

Paramètre	Type	Définition
noSeat	int	Numéro de siège du joueur.

Prise de décision :
Envoyé lorsque le joueur doit prendre une décision.
Syntaxe := TAKE_ACTION;*checkAllowed*;*foldAllowed*;*callAllowed*;*callAmount*;
raiseAllowed;*minRaiseAmount*;*maxRaiseAmount*;

Paramètre	Type	Définition
checkAllowed	bool	Vrai si le joueur peut parler (*Check*).
foldAllowed	bool	Vrai si le joueur peut se coucher (*Fold*).
callAllowed	bool	Vrai si le joueur peut suivre (*Call*).
callAmount	int	Le nombre de jetons nécessaire pour suivre.
raiseAllowed	bool	Vrai si le joueur peut augmenter la mise (*Raise*).
minRaiseAmount	int	Le nombre de jetons minimums du *Raise*.
maxRaiseAmount	int	Le nombre de jetons maximums du *Raise*.

Fin d'une action :
Envoyé lorsqu'un joueur a pris une décision.
Syntaxe := PLAYER_TURN_ENDED;*noSeat*;*betAmount*;*moneyAmount*;
totalPotAmount;*action*;*actionAmount*;

Paramètre	Type	Définition
noSeat	int	Numéro de siège du joueur.
betAmount	int	La mise courante du joueur.
moneyAmount	int	Le montant en banque du joueur.
totalPotAmount	int	Le nombre de jetons total sur la table.
action	int	Identificateur de l'action du joueur.
actionAmount	int	Le montant associé à l'action du joueur.

Fin d'un tour de mise :
Envoyé à la fin de chaque tour de mise (*preflop*, *flop*, *turn* et *river*).
Syntaxe := BETTING_TURN_ENDED; *potsAmount*
potsAmount :=*potAmount;* (répéter 9 fois)

Paramètre	Type	Définition
potAmount	int	Nombre de jetons de chaque pot.

Changement de carte d'un joueur :
Envoyé lorsqu'un joueur reçoit ou dévoile ses cartes.
Syntaxe := PLAYER_CARD_CHANGED;*noSeat*;*idCard1*;*idCard2*;

Paramètre	Type	Définition
noSeat	int	Numéro de siège du joueur.
idCard1	int	Identificateur de la carte 1 du joueur.
idCard2	int	Identificateur de la carte 2 du joueur.

Changement des cartes du plateau :
Envoyé lorsque l'on montre les cartes du *Flop*, de la *Turn* et de la *River*.
Syntaxe := BOARD_CHANGED;*boardCards*
boardCards :=*boardCard;* (répéter 5 fois)

Paramètre	Type	Définition
boardCard	int	Identificateur d'une carte du plateau.

Pot remporté :
Envoyé lorsqu'un joueur remporte un pot.
Syntaxe := POT_WON;*noSeat*;*idPot*;*potAmountWon*;*moneyAmount*;

Paramètre	Type	Définition
noSeat	int	Numéro de siège du joueur.
idPot	int	L'index du pot remporté.
potAmountWon	int	Le montant remporté.
moneyAmount	int	Nouveau montant en banque du joueur.

Fin de partie :
Envoyé à la fin d'une partie.
Syntaxe := GAME_ENDED;

Modification de la banque d'un joueur:
Envoyé lorsqu'un joueur ajoute ou retire un montant de sa banque.
Syntaxe := PLAYER_MONEY_CHANGED;*noSeat*;*moneyAmount;*

Paramètre	Type	Définition
noSeat	int	Numéro de siège du joueur.
moneyAmount	int	Nouveau montant en banque.

Nouveau joueur :
Envoyé lorsqu'un joueur se joint à la partie.
Syntaxe := PLAYER_JOIN;*noSeat*;*playerName*;*moneyAmount*;

Paramètre	Type	Définition
noSeat	int	Numéro de siège du joueur.
playerName	String	Nom du joueur.
moneyAmount	int	Montant de la banque du joueur.

Un joueur quitte la table :
Envoyé lorsqu'un joueur quitte la table.
Syntaxe := PLAYER_LEFT;*noSeat*;

Paramètre	Type	Définition
noSeat	int	Numéro de siège du joueur.

En attente de joueur :
Envoyé lorsqu'il n'y a pas assez de joueurs pour commencer une partie.
Syntaxe := WAIT_FOR_PLAYER;

Ping :
Envoyé lorsque le serveur veut vérifier si le client est toujours vivant.
Syntaxe := PING

Fermeture de la table :
Envoyé lorsque la table est détruite.
Syntaxe := TABLE_CLOSED;

Communication du client à la table

Décision :
Le client envoie son action au serveur lorsque le serveur lui demande de prendre une décision.
Syntaxe := CHECK; |
CALL; |
FOLD; |
RAISE;*raiseAmount*;

Paramètre	Type	Définition
raiseAmount	int	Montant de la mise.

Communication du lobby au serveur de poker

Connection :
Envoyé par le lobby lorsqu'il amorce la communication avec le serveur.
Syntaxe := AUTHENTIFICATION;*playerName*
Message de retour := *success*;

Paramètre	Type	Définition
playerName	String	Nom du joueur.
success	bool	Vrai si le joueur est correctement authentifié.

Créer une table :
Envoyé pour créer une nouvelle table sur le serveur.
Syntaxe := CREATE_TABLE;*tableName*;*gameType*;*bigBlind*;*nbSeats*;
Message de retour := *port*;

Paramètre	Type	Définition
tableName	String	Nom de la table
gameType		Variante du Texas Hold'Em
bigBlind	int	Montant du *big blind*
nbSeats	int	Le nombre maximum de joueurs sur la table.
port	int	Le numéro de port de la table. Le port vaut -1 si la table n'a pas été créée.

Liste les tables existantes :
Envoyé pour obtenir la liste des tables.
Syntaxe := LIST_TABLES;
Message de retour := *port*;*tableName*;*gameType*;*bigBlind*;*nbPlayers*;*nbSeat*;

Paramètre	Type	Définition
port	int	Le numéro de port de la table. Le port vaut -1 si la table n'a pas été créée.
tableName	String	Nom de la table
gameType		Variante du Texas Hold'Em
bigBlind	int	Montant du *big blind*
nbSeats	int	Le nombre maximum de joueurs sur la table.

Déconnection :
Envoyé lorsque le client quitte le serveur.
Syntaxe := DISCONNECT;

Communication du lobby à la table

Connection :
Envoyé par le lobby lorsqu'il amorce la communication avec la table.
Syntaxe := AUTHENTIFICATION;*playerName*
Message de retour := *success*;

Paramètre	Type	Définition
playerName	String	Nom du joueur.
Success	bool	Vrai si le joueur est correctement authentifié.

Joindre une table :
Envoyé par le lobby pour s'asseoir à la table. Le client doit s'être authentifié avant de pouvoir joindre la table.
Syntaxe := JOIN_TABLE;
Message de retour := *noSeat*;

Paramètre	Type	Définition
noSeat	int	Numéro de siège du joueur. -1 si le joueur n'a pas réussi à s'asseoir.