UNIVERSITÉ DE LORRAINE | UFR MATHÉMATIQUES INFORMATIQUE MÉCANIQUE ET AUTOMATIQUE

Projet en Fouille de données

Master 1 informatique

Joueur artificiel d'Awélé

⇨ **Réalisé par :**

- **Hani SLAIM**
- **Ahmed ZEKRI**

Sommaire :

I. Introduction

Dans la vie courante, chaque jour, on est confronté à de nombreux cas, où on se trouve obligé de prendre des décisions. Cependant, le fait de trouver les bons raisonnements ou les bonnes stratégies pour prendre les bonnes décisions n'est pas toujours évident. Par exemple, en économie ou lors d'un jeu stratégique, on est contraint d'étudier le comportement des différents acteurs. On cherche dans les deux cas à maximiser les gains et à minimiser les pertes.

C'est vrai que tout le monde prend habituellement des décisions immédiates, mais assez souvent, en se trouvant confronté à des situations complexes, ou même à défaut de présence psychologique ou mentale, il arrive qu'on se trempe. Parfois, même si on demande l'avis de proches ou d'experts, la prise de décision efficace n'est pas toujours assurée.

Heureusement que de nos jours, le développement des systèmes informatisés qui décident efficacement à notre place est de plus en plus répandu. Ces systèmes doivent être efficacement conçus afin de permettre de trouver les meilleures solutions pour chaque cas selon des connaissances bien acquises. Mais arriver à implémenter ces systèmes experts n'est pas de même une tâche facile.

Dans le domaine de divertissement, la prise de décision est développée essentiellement dans les agents artificiels des jeux stratégiques.

En fait, le jeu d'Awelé semble être un excellent cas pratique d'étude pour concevoir un prototype simple mais efficace d'un système expert qui utilise certaines techniques de fouilles de données pour gagner les parties du jeu.

On va voir dans ce qui suit donc comment on est arrivé à concevoir cet agent artificiel qu'on a appelé « Slaim_Zekri ».

II. Présentation Générale du projet

1. Cadre

Ce rapport porte sur les phases de développement du projet de fouille de données du groupe constitué de : Hani SLAIM et Ahmed ZEKRI. Etant étudiants en première année Master informatique à l'Université de Lorraine (site de Metz), et souhaitant poursuivre nos études respectivement en IHM et ID, nous avons tous les deux immédiatement choisi avec un grand intérêt cette option qui présente une discipline bien importante pour ces deux parcours.

Le travail consiste à développer un joueur artificiel du jeu d'Awelé. Pour ce faire, on a utilisé l'outil Rstudio pour le développer en langage R. Notre professeur M. Alexandre Blansche nous a déjà donné des classes prédéfinies du jeu et nous ne sommes imposés de réaliser qu'un joueur artificiel (computer player) qui pourra battre ceux développés par les autres groupes.

2. Sujet

a) Travail demandé :

Le sujet de ce projet consiste à construire un moteur d'intelligence artificielle permettant d'implémenter un joueur à l'Awélé, en utilisant un ou plusieurs algorithmes de fouille de données. L'objectif est donc de construire un modèle de prise de décision permettant de choisir le plus efficacement possible le meilleur coup à jouer en fonction d'une situation donnée. La solution proposée utilisera un algorithme de fouille de données afin d'extraire les connaissances à partir d'une base de données de coups joués, et utilisera ces connaissances pour prendre la décision.

L'Awélé est un jeu traditionnel africain. Le jeu connaît de nombreuses appellations et de nombreuses variantes. C'est un jeu de stratégie combinatoire abstrait qui voit s'affronter deux joueurs jouant à tour de rôle, sans informations cachées et sans hasard. Ce projet considère la variante la plus utilisée du jeu.

Il est demandé de réaliser, en langage de programmation R, un « bot » (joueur artificiel) permettant de jouer à l'Awélé. Toutes les fonctions permettant de gérer le bot devront se trouver dans un fichier bot.nom_choisi.R et les noms de fonction et des variables devront être préfixés par le nom du bot. Le bot en en lui-même devra être une fonction nommée nom_choisi et sera passé en paramètre à la fonction awele.match. On pourra s'inspirer des fichiers bot.random.R et bot.nb.R donnés. L'affrontement entre deux bots se fera en suivant le modèle du ficher match.nb.vs.random.R.

b) Données :

On dispose d'un ensemble de données correspondant à 303 situations de jeu observées au cours de plusieurs parties entre un joueur expérimenté et un joueur novice. Le joueur expérimenté a remporté toutes les parties. Chaque observation est décrite par 14 variables. Les 12 premières correspondent à la situation du plateau de jeu (nombre de graines dans les trous, de gauche à droite pour le joueur actif et de droite à gauche pour l'adversaire). La 13e variable est la décision prise par le joueur actif. La dernière variable indique si le joueur a gagné ou perdu la partie.

Le fichier awele.R contient un moteur de jeu d'Awélé permettant de voir s'affronter deux bots. La classe awele.class est une liste composée de trois champs (variable qui présente nombre de graines dans chaque plateau, vecteur du joueur actifs et de son adversaire, et un autre vecteur représentant leurs scores). La fonction awele.match fait s'affronter deux bots passés en paramètres. La fonction fait jouer deux matchs en changeant de premier joueur. Pour chaque match, le résultat est affiché (vainqueur et score). Les deux paramètres bot1 et bot2 doivent être des fonctions qui prennent en paramètre un objet de classe awele.class et qui retourne un vecteur de six valeurs numériques correspondant à l'efficacité supposée de chacun des six coups possibles.

III. Préparation

1. Recherche

Pour chercher à trouver des idées qui nous aident à trouver des solutions sur ce sujet, on a essayé au début de bien comprendre le jeu (que nous deux ne connaissons pas auparavant).

On a donc regardé quelques vidéos (sur YouTube) qui mettent l'accent sur la manière du jeu d'Awelé et de ses différentes stratégies gagnantes.

Ensuite, on a consulté le livre "Stratégies des joueurs d'Awalé" de Jean Retscitzki, et plus précisément la partie au quelle il parle des stratégies et les règles à suivre pour gagner à coup sûr. Ce bouquin a divisé ces stratégies en trois sous parties :

- **les stratégies de début de partie** : il est conseillé de jouer des cases non consécutives (alterner) de son territoire durant les premiers tours de la partie afin d'empêcher les captures multiples de l'adversaire par la suite du jeu.
- **Les stratégies de milieu de partie** : Ne pas se contenter d'attaquer et assurer une bonne défense. Augmenter le contenu des cases menaçantes de son adversaire (savoir semer pour récolter) : une série de cases vides de l'adversaire, judicieusement ensemencée, peut se transformer en récolte multiple fort intéressante. Accumulation d'un nombre de graines suffisant pour faire un tour complet (soit au moins 12 graines) dans sa case. Cette stratégie s'appelle « krou » et plusieurs techniques défensives permettent d'en contrecarrer (Soit par le **blocage** qui s'explique par le rassemblement d'au moins 2 graines dans la case menacée par le Krou ; soit par la **surcharge** qui consiste à ajouter des graines au Krou de l'adversaire afin de le surcharger et ainsi lui faire manquer sa cible ; soit par la **privation** qui consiste à ne pas fournir de graines à l'adversaire, afin de l'obliger à jouer son Krou avant qu'il ne soit mûr ou soit par la **contre-attaque** qui consiste à accumuler des graines dans une ou plusieurs cases afin de récolter également quand le Krou de l'adversaire sera joué.)
- **Les stratégies de fin de partie** : Ces stratégies sont valables une fois le plateau est presque vide. Le joueur peut donc construire des "pièges" en privant l'adversaire de graine, puis de lui en fournir une qu'il sera obligé de jouer sur une case menacée. Enfin, le joueur peut garder les graines dans son camp afin d'affamer l'adversaire, et ainsi se mettre volontairement dans l'impossibilité de le nourrir lorsque celui-ci n'a plus de graine.

On a enfin téléchargé un jeu Android qui s'appelle « Oware » sur smartphone afin d'appliquer ce qu'on a appris et surtout pour mettre en œuvre les différents stratégies qu'on vient de citer. C'était très amusant ! Et ce jeu est devenu notre préféré ☺

2. Idées et réflexions

a) Idée 1 :

Choisir une technique de fouille de données (LR, NB, LDA, CDA, SVM, ...) en fonction de celle utilisée de l'adversaire

⇨ Chercher le bot adversaire dans la source ("bot...') dans "match.nb.vs..."
⇨ Chercher la technique utilisée par l'adversaire dans le fichier "bot…"
⇨ Selon la technique trouvée (utilisée par l'adversaire), utiliser celle gagnante (Après un test de plusieurs techniques sur chacune donnée).

b) Idée 2 :

Utiliser un autre algorithme non vu en cours une fois on l'a trouvé plus efficace.

c) Idée 3 :

Utiliser l'un des algorithmes étudiés en cours, et on ajoute des données pour qu'il soit efficace.

d) Idée 4 :

Utiliser simultanément plusieurs algorithmes de fouille de données étudiés en cours pour combiner leur performance.

e) Idée 5 :

Combiner la méthode 3 avec 4 en réduisant le nombre de données. Car en effet les données peuvent parfois parasiter ou faire du bruit, ce qui implique les impressions de certains algorithmes.

IV. Réalisation

1. Essais

On a essayé au début d'implémenter les idées qu'on a donné précédemment dans notre solution, une par une.

a) Essai de l'idée 1 :

On a réalisé une fonction qu'on a appelé (**techAdv**), qui retourne le modèle selon la technique de fouille de donnée utilisée par le bot adversaire.

Cette méthode suit le principe suivant :

- Définition d'un compteur (variable d'environnement global) qui va nous servir comme le numéro de la ligne courante du match entre notre bot et un adversaire
- Recherche du fichier des matchs
- Extraire le nom du bot adversaire en parcourant la source du fichier des matchs pour chercher toutes les occurrences de matchs de notre bot (en ignorant les indices des lignes de matchs en commentaires)
- Chercher la technique utilisée dans le fichier du bot de l'adversaire et selon cette technique retrouver la meilleure (qui lui bat le plus selon nos tests)
- Retourner le modèle choisi

Après le test de plusieurs bots qu'on a implémenté chacun utilisant des méthodes simples des approches de fouille de données vues au cours (NB, ADL, AFD, RL, KNN, CART, MLP, SVM) et qui sont implémentés dans la librairie fdm2id. Les tests qu'on a effectués nous ont donné les résultats suivants :

Technique utilisée par le bot l'adversaire	Meilleure technique pour le gagner
NB	SVM
ADL	CART
AFD	MLP
RL	KNN
KNN	CART
CART	MLP
MLP	CART
SVM	KNN

Malgré la performance de cette méthode, on a décidé finalement d'abandonner son usage, vu qu'on a trouvé quelques limites comme l'usage de noms des fichiers de bots non conformes aux noms utilisés à l'implémentation. De plus, cette technique ne pourra marcher que dans certains cas simples où l'adversaire utilise une des approches de fouille de données qu'on a testés.

b) Essai de l'idée 2 :

En cherchant sur des ouvrages sur internet, on a trouvé un bon document qui traite particulièrement quelques algorithmes d'évaluation et de classification pour le jeu d'Awelé : (https://www.scei-concours.fr/tipe/TIPE_2012/sujets_2012/INFORMATIQUE_MP.pdf)

Après quelques essais de développement de ces algorithmes (MiniMax, NegaMax et Élagage Alpha-Beta), on a abandonné encore cette idée vu le temps et la complexité de leur développement.

c) Essai de l'idée 3 :

On a essayé dans cette idée de cumuler un ensemble de méthodes qui nous permettent d'améliorer au maximum possible notre base d'apprentissage (utilisation des données et les transformer en informations, et parcours de l'espace de recherche du plateau du jeu courant à différentes reprises) afin de créer les situations du coup idéal tout en prenant compte des différentes techniques de défense et d'attaque.

Parmi les stratégies, qu'on a mises en œuvre, on peut citer :

- Le contrôle des cases vides du camp adverse.
- La construction et la détention des blocages (bloquer les cases vides adverses)
- Le compte du nombre de coups que le joueur peut jouer à l'intérieur de son propre camp, sans donner de graine a l'adversaire.

Ceci nous a mené à introduire de nouvelles fonctions dans notre bot, qui permettent en fait d'ajouter les nombres suivants dans la base du jeu :

- Graines de chaque joueur
- Nos cases vides et celles de l'adversaire
- Cases qui n'ont qu'une ou deux graines
- Graines gagnées en fonction de la case jouée

d) Essai de l'idée 4 :

On a donc développé une nouvelle technique qui permet de choisir un modèle utilisant la meilleure technique parmi trois algorithmes de fouille de données étudiés en cours afin de combiner leur performance. En fait, on a commencé par la création de nos trois modèles, chacun utilisant l'une des techniques (LDA, LR, SVM) comme technique de classification. Lors de chaque création de modèle, on sélectionne seulement les coups joués par le vainqueur des confrontations et les cases où on a une ou deux graines pour le coup suivant afin rendre plus efficace et plus rapide nos algorithmes. Ensuite, on détermine la meilleure prédiction de gain entre les trois techniques données à l'état actuel du plateau en cherchant bien évidemment le plus grand gain.

e) Essai de l'idée 5 :

Cette idée sera décrite en détails dans la section suivante.

2. Solution retenue

⇨ Le principe général de fonctionnement de notre bot est le suivant :
- Sélectionner toutes les observations correspondantes aux coups joués par le vainqueur de la partie.
- Sélectionner dans la base que les linges qui ont moins 3 case vides : on testant plusieurs réduction de données possible nous nous somme rendu compte que réduire les données selon le nombre de case vides que le joueur détient augmente considérablement la performance des prédictions et arrive à battre la plupart des bots.
- Construire trois modèles (LDA, LR, SVM) à partir des 12 premières variables de ces observations : nous avons testé plusieurs triplets d'algorithme et celui qui donne la meilleure performance est le triplet (LDA, LR, SVM). Nous avons choisi un triplet car au delà de 3 algorithmes le bot devient plus au moins lourd à charger.
- Prédire la 13ème (le coup joué).

⇨ Notre bot nécessite au début le chargement de la bibliothèque fdm2id (disposant de toutes les techniques de classification et de FD) et les données de la base (awele.data) (fournies par le prof).

⇨ Voici les fonctions implémentées dans notre bot :
- **Slaim_Zekri :** Fonction qui permet juste d'exécuter Slaim_Zekri.exec (en utilisant nos trois modèles Slaim_Zekri.model (1,2 et 3) et en fonction de l'instance de la classe awele).
  - **Slaim_Zekri.exec** : Fonction d'évaluation de la meilleure solution selon l'état du plateau de jeu et des trois modèles. Elle prend en paramètre la base du jeu en plus des trois modèles créés. On récupère l'état du plateau de jeu (sous la forme d'une matrice plutôt que d'un vecteur) et on ajoute les données après exécution de la fonction ajouterLesDonnees qu'on expliquera plus tard. Puis, on modifie les noms des colonnes pour correspondre aux noms dans l'ensemble d'apprentissage. Après, on applique les trois modèles et on retourne les prédictions (sous forme de degrés d'appartenance aux classes). On utilise donc la fonction predict (du package arules) en donnant à chaque fois l'un des trois modèle comme premier paramètre, puis une instance du jeu courant g, et fuzzy = Faux (pour retourner un seul résultat). Une fois, on a le résultat de chaque prédiction, on utilise la fonction meilleurPredict pour choisir la meilleure entre eux. On crée enfin une nouvelle dataframe pour y coller les noms des colonnes du résultat de la meilleure prédiction.}
    - **meuilleurPredict** : cette fonction permet de déterminer la meilleure prédiction de gain entre deux données à un état donné du plateau. L'index renvoyé de la case où on peut avoir le meilleur gain, est associé à un nom qui est retourné par la fonction nomCaseJouer.
      - **nomCaseJouer** : Définition d'un nom d'une case jouée selon une position donnée (1..6)
  - Construction des trois modèles (**Slaim_Zekri.create.model** (**1**, **2**, et **3)** à partir de la base (awele.data) donnée)
    - Utilisation de trois fonctions de construction du modèle qui exécutent la fonction ajouterLesDonnees, et qui sélectionnent ensuite les instances correspondantes aux coups joués par le vainqueur des affrontements et aux lignes ayant moins de trois cases vides dans le plateau du joueur, pour utiliser l'une des 3 techniques suivantes :
      - **Slaim_Zekri.create.model1 :** construction du modèle de classification avec l'algorithme LDA.

- **Slaim_Zekri.create.model2** : construction du modèle de classification avec l'algorithme LR.
- **Slaim_Zekri.create.model3** : construction du modèle de classification avec l'algorithme SVM.

- **ajouterLesDonnees** : Une fonction qui rajoute des données à la base actuelle : Création d'un nouveau frame avec les colonnes supplémentaires, Recopie des colonnes de base, mise à jour des données de la base en exécutant les fonctions (ajouterSomme, Ajoutervide, SommePetitGrain, leGagneParCase avec une position d'ajout donnée)
  - **ajouterSomme** : Ajout du nombre de graines de chaque joueur dans la base
  - **Ajoutervide** : Ajout du nombre des cases vides pour le joueur et l'adversaire dans la base
  - **SommePetitGrain** : Ajout du nombre de cases qui ont une ou deux graines
  - **leGagneParCase** : Calcul du nombre de graines gagnées si la case est jouée : {Parcours des lignes et des colonnes de la base de données de distribution en vidant la case courante et en distribuant ses billes aux prochaines cases (décrémentation du nombre de billes et mise à jour de la position courante) jusqu'à l'arrivée à la case au quelle s'arrête la dernière bille. A partir de cette case et de ses précédentes sera calculé le nombre de billes gagnés (en vérifiant ainsi l'appartenance de la case au camp de l'adversaire et de même pour les cases qui la précédent).}

3. Jeu de tests :

```
Matchs avec nb1
[1] "Slaim_Zekri (26 - 2)"
[1] "Slaim_Zekri (26 - 6)"
[1] "Slaim_Zekri"
[1] "Slaim_Zekri (26 - 6)"
[1] "Slaim_Zekri (26 - 2)"
[1] "Slaim_Zekri"

Matchs avec nb2
[1] "Slaim_Zekri (29 - 15)"
[1] "Slaim_Zekri (28 - 4)"
[1] "Slaim_Zekri"
[1] "Slaim_Zekri (28 - 4)"
[1] "Slaim_Zekri (29 - 15)"
[1] "Slaim_Zekri"

Matchs avec random
[1] "Slaim_Zekri (27 - 9)"
[1] "Slaim_Zekri (25 - 11)"
[1] "Slaim_Zekri"
[1] "Slaim_Zekri (24 - 18)"
[1] "Slaim_Zekri (28 - 9)"
[1] "Slaim_Zekri"
```

```
Matchs avec adl
[1] "Slaim_Zekri (29 - 0)"
[1] "Slaim_Zekri (27 - 12)"
[1] "Slaim_Zekri"
[1] "Slaim_Zekri (27 - 12)"
[1] "Slaim_Zekri (29 - 0)"
[1] "Slaim_Zekri"

Matchs avec afd
[1] "Slaim_Zekri (25 - 5)"
[1] "Slaim_Zekri (25 - 3)"
[1] "Slaim_Zekri"
[1] "Slaim_Zekri (25 - 3)"
[1] "Slaim_Zekri (25 - 5)"
[1] "Slaim_Zekri"

Matchs avec rl
[1] "Slaim_Zekri (26 - 17)"
[1] "Slaim_Zekri (25 - 15)"
[1] "Slaim_Zekri"
[1] "Slaim_Zekri (25 - 15)"
[1] "Slaim_Zekri (26 - 17)"
[1] "Slaim_Zekri"
```

```
Matchs avec knn
[1] "knn (26 - 7)"
[1] "Slaim_Zekri (25 - 0)"
[1] "Slaim_Zekri"
[1] "Slaim_Zekri (25 - 0)"
[1] "knn (26 - 7)"
[1] "Slaim_Zekri"

Matchs avec cart
[1] "cart (25 - 17)"
[1] "Slaim_Zekri (27 - 11)"
[1] "Slaim_Zekri"
[1] "Slaim_Zekri (27 - 11)"
[1] "cart (25 - 17)"
[1] "Slaim_Zekri"

Matchs avec mlp
[1] "Slaim_Zekri (31 - 2)"
[1] "Slaim_Zekri (27 - 12)"
[1] "Slaim_Zekri"
[1] "Slaim_Zekri (27 - 12)"
[1] "Slaim_Zekri (31 - 2)"
[1] "Slaim_Zekri"
```

V. Conclusion

Tout au long de ce rapport, on a mis l'accent sur les phases de développement de notre joueur artificiel d' « Awelé ». Ce jeu de stratégie traditionnel africain est caractérisé par ses nombreux aspects de réflexions mentales et mathématiques, qu'on a pu en implémenter certains dans notre solution. On a expliqué en fait les techniques ainsi que les stratégies qu'on a utilisées afin d'arriver à créer un joueur automatique qu'on a jugé très performant après plusieurs jeux de test effectués. On a noté surtout la performance des algorithmes LDA, LR, et SVM pour notre cas pratique bien particulier. Ceci nous a montré la grande utilité des techniques de fouille de données dans notre vie habituelle, notamment dans le cadre du cas pratique simple étudié.

Ce projet a été, bel et bien une nouvelle opportunité d'étude pour notre groupe. Nous avons eu l'occasion d'aller chercher, comprendre, et utiliser de nouvelles notions que nous avons bien aimé les découvrir et dont nous sommes actuellement plus impressionnés que jamais. Nous avons pu constater la grande importance de l'application des algorithmes de fouille de données dans certains problèmes à caractère décisionnels et l'apport extraordinaire que cette discipline pourra invoquer.

VI. Références

- *"Stratégies des joueurs d'Awalé", Jean Retschitzki, L'Harmattan 1990*
- https://www.scei-concours.fr/tipe/TIPE_2012/sujets_2012/INFORMATIQUE_MP.pdf
- http://abstractstrategygames.blogspot.fr/2010/10/awale.html
- http://s.helan.free.fr/awale/conseils/
- http://www.nongnu.org/awale/tutorial_fr.html
- https://www.scei-concours.fr/tipe/TIPE_2012/sujets_2012/INFORMATIQUE_MP.pdf
- http://www.strategie-aims.com/events/conferences/6-xviieme-conference-de-l-aims/communications/1708-les-apports-de-la-theorie-des-jeux-au-management-strategique/download
- http://www.myriad-online.com/resources/docs/awale/francais/printable.htm