Une expérience d'élicitation de connaissances expertes dans le domaine du Bridge

Colin Deheeger*, Jean Pierre Desmoulins*, Jean Baptiste Fantun*, Swann Legras*
Alexis Rimbaud*, Céline Rouveirol*,**, Henry Soldano*,**, Véronique Ventos*

*NukkAI, Paris, France (www.nukk.ai)
cdeheeger@nukk.ai, jpdesmoulins@nukk.ai, jbfantun@nukk.ai, slegras@nukk.ai
arimbaud@nukk.ai, crouveirol@nukk.ai, henry.soldano@nukk.ai, vventos@nukk.ai
**LIPN, Université Paris 13, Villetaneuse, France
rouveirol@lipn.univ-paris13.fr, henry.soldano@lipn.univ-paris13.fr

Résumé. Nous cherchons à construire une représentation de la décision à prendre lors d'une phase des enchères au jeu de bridge. Cela suppose plusieurs étapes. Il faut d'abord construire un échantillon des situations auxquelles on peut être confronté lors de cette phase. À chacune de ces situations, on associe la décision adéquate, en effectuant de multiples simulations avec un solveur boîte noire. Enfin, par apprentissage supervisé relationnel, et étant donné un vocabulaire choisi, on obtient un modèle interprétable expliquant la décision. Le succès de cette méthodologie suppose un certain nombre de choix et d'interactions entre résultats produits et experts du jeu que nous décrivons dans cet article.

1 Introduction

Notre but est de modéliser des processus de décision experts au bridge et, pour ce faire, nous mettons en place une méthodologie réutilisable faisant intervenir à la fois des experts humains, des programmes de décision boîte noires, et des systèmes d'apprentissage automatique supervisé relationnel. On souhaite *in fine* obtenir un modèle explicite avec de bonnes performances prédictives. On souhaite également pouvoir expliquer la décision associé à un cas donné, par opposition aux méthodes de résolution ou d'apprentissage de type boîte noire. Suite aux succès des méthodes supervisées de la famille des réseaux profonds d'une part et à une pression grandissante d'une société civile (loi RGPD) imposant que la transparence soit faite autour de processus de décision automatiques nous affectant à divers niveaux, de plus en plus de chercheurs en IA (ré-)explorent des techniques permettant d'intepréter, de justifier ou d'expliquer des classifieurs. Lorsque ces classifieurs sont de type boîte noire il s'agit d'apprendre a posteriori des modèles explicites dans des langages symboliques, le plus souvent sous forme de règles ou d'arbres de décision (Ribeiro et al., 2016). Ces modèles peuvent également être appris directement sous forme de représentations explicites. En effet, en plus de leur propre performance en terme de prédiction, les modèles

explicites extraits par apprentissage supervisé peuvent être porteur de connaissances expertes (Murdoch et al., 2019), qui ont une valeur intrinsèque, en terme d'explicabilité, de pédagogie, et d'évaluation en terme d'éthique ou d'équité (parce qu'ils peuvent permettre d'expliciter des biais liés au système d'apprentissage ou à l'ensemble des exemples d'entraînement). Le plus souvent, les travaux préconisent une méthode en deux étapes : une première a pour but de construire un classifieur aussi précis que possible, une deuxième étape permet de générer à la demande un ensemble de règles explicites et locales pour justifier de la classification d'un exemple.

Nous présentons ici une méthodologie complète d'acquisition d'un tel ensemble de règles. Nous avons choisi d'attaquer directement le problème d'apprentissage supervisé avec des méthodes d'apprentissage supervisé relationnel, relevant précisément de la *Programmation Logique Inductive* (ou PLI). Le langage de règles de ces méthodes, une restriction de la logique d'ordre un, permet d'apprendre des règles compactes et sous une forme relativement compréhensible par des experts du domaine. Ce langage permet également d'utiliser le vocabulaire du domaine, défini dans une théorie du domaine, comme illustré dans de premiers travaux sur le sujet (Legras et al., 2018).

Le plan de l'article est le suivant. Après une brève introduction sur le jeu de bridge (section 2), nous décrivons en section 3 une modélisation originale qui a pour but de générer des exemples relationnels pour le problème d'apprentissage cible. Nous décrivons ensuite brièvement dans la section 4 les systèmes de PLI utilisés, ainsi que leurs paramètres, puis les expériences menées (section 5). Nous consacrons la section 6 à une analyse des modèles obtenus, en particulier pour tout ce qui concerne l'interpétabilité des règles / arbres de décision construits. Des pistes de recherche sont esquissées en conclusion.

2 Le problème étudié

2.1 Brève introduction sur le bridge

Le bridge se pratique à 2 (« Nord » et « Sud ») contre 2 (« Est » et « Ouest ») avec un jeu classique de 52 cartes, chaque joueur recevant en début de partie une main aléatoire de $\frac{52}{4} = 13$ cartes. L'objectif de chaque camp est d'optimiser un score qui dépend :

— **de la vulnérabilité** de chaque camp, paramètre binaire extérieur influençant la valeur des scores.

— **du contrat** appelé lors de la première phase du jeu, les enchères. Un contrat est l'engagement d'un camp à réaliser un minimum de levées $l_{min} \in \{1,\dots 7\}$ à un atout donné (♣, ♦, ♥ ou ♠) ou à Sans-Atout (« NT » pour « No Trump »). Il peut être contré, l'adversaire imposant ainsi d'augmenter le score final. Il est noté pS (ou pS^X s'il est contré), $p = l_{min} - 6 \in \{1,\dots 7\}$ étant le « palier » et $S \in \{♣, ♦, ♥, ♠, NT\}$.

— **du nombre de levées réalisées** par ce camp pendant la seconde phase, le « jeu de la carte ».

Demander et gagner un contrat au palier de 6 ou 7 permet de gagner une prime. Pour plus de précisions sur le bridge, le lecteur pourra consulter (Université du Bridge, 2013; Féderation Française de Bridge, 2018; ACBL, 2019). Deux concepts sont essentiels pour le travail présenté ici :

— **Les enchères** : Elles permettent à chaque joueur (le premier s'appelant le « donneur ») qui le souhaite de divulguer des renseignements codés sur son jeu, ce code étant public mais en pratique ni correct ni complet. Chacun déclare à tour de rôle dans le sens des aiguilles d'une montre, les joueurs pouvant utiliser comme éléments de langage : « Passe », « Contre » ou un contrat supérieur au précédent nommé (♣ < ♦ < ♥ < ♠ < NT à chaque palier). Le dernier contrat nommé, suivi de trois « Passe », est celui qui doit être joué.

— **L'évaluation de la force d'un jeu** : Les bridgeurs attribuent une valeur aux plus grosses cartes : un As vaut 4 HCP (pour « High Card Points »), un Roi 3 HCP, une Dame 2 HCP et un Valet 1 HCP. Les renseignements donnés par les joueurs sur leur main portent sur son nombre de points HCP et sa distribution (nombre de cartes dans une ou plusieurs couleurs).

2.2 Position du problème

Les experts décrivent la situation suivante :

— Ouest, donneur, enchérit 4♠, ce qui signifie approximativement qu'il a un minimum de 7 cartes à Pique et un maximum de 10 HCP dans son jeu.

— Nord contre ce qui signifie approximativement qu'il possède un minimum de 13 HCP et, à moins d'avoir un jeu très fort, qu'il a un minimum de trois cartes dans chacune des autres couleurs (♣, ♦ et ♥).

— Est passe, ce qui n'a pas de signification particulière.

Sud doit alors prendre une décision : passer ou proposer un contrat dans son camp. Cette décision est potentiellement très coûteuse dans une partie de bridge mais la situation est relativement rare, aussi les experts n'ont-ils pas de connaissances précises sur ce sujet. L'objectif est de développer une méthodologie pour fournir une représentation de la décision à prendre par Sud, ce qui nécessite d'abord de disposer de données étiquetées que ne peuvent fournir les experts. Il est à noter que Derek Patterson s'est intéressé à la résolution de ce problème par algorithmes génétiques (Patterson, 2008).

2.3 Méthodologie de génération automatique des données

2.3.1 Vue générale

Cette méthodologie, décrite dans ce qui suit, est la suivante :

— Modélisation complète du problème

— Génération automatique des données d'apprentissage et de test

— Étiquetage automatique de ces données

Ces données étiquetées seront utilisées pour l'induction de règles (Aleph) et d'arbres de décision (Tilde) proposées à la validation des experts.

2.3.2 Modélisation du problème

Les bridgeurs n'explicitant qu'approximativement les caractéristiques de leurs enchères, les experts ont modélisé de manière correcte et complète les mains correspondant à l'enchère de 4♠ et au Contre en vue d'une génération automatique des données.

Dans la suite de cette section,

— « distribution exacte $nmpq$ » signifie que la main comporte n cartes à ♠, m cartes à ♥, p cartes à ♦ et q cartes à ♣ avec toujours $n+m+p+q=13$.
— « distribution $n'm'p'q'$ » est la distribution exacte par ordre décroissant.
— $|c|$ est le nombre de cartes détenues dans la couleur $c \in \{♠, ♥, ♦, ♣\}$.

Modélisation de l'enchère de 4♠ Les experts ont modélisé l'enchère de 4♠ par 17 règles $4♠ \leftarrow (R_0, R_i)_{1 \leq i \leq 17}$, R_0 étant une prémisse commune indépendante de la vulnérabilité de chaque camp et R_i en dépendant. En voici deux exemples :

— R_2 - Non vulnérable contre vulnérable, |♥|=1, 2 cartes exactement parmi As, Roi, Dame et Valet de ♠, une distribution 7321.
— R_5 - Non vulnérable contre vulnérable, (|♣| ≥ 4 ou |♦| ≥ 4), 2 cartes exactement parmi As, Roi, Dame et Valet de ♠, une distribution 7mpq avec $m \geq 4$.

La mise au point de ces règles a été faite par des échanges avec les experts sur des ensembles de donnes tirées au hasard et filtrées pour que la main d'Ouest vérifie au moins une des règles. Lors de la validation finale, 8 200 000 donnes ont été tirées et 10 105 d'entre elles possédaient une main d'Ouest vérifiant l'une des 17 règles précédentes relatives à l'enchère de 4♠. Aucune règle n'a un support vide, R_2 possède par exemple un support de 16.2% et R_5 de 15%.

Modélisation du Contre De même, les experts du bridge ont modélisé le Contre par trois règles $\text{Contre} \leftarrow (R'_0, R'_i)_{1 \leq i \leq 3}$, R'_0étant indépendante de la vulnérabilité et R'_i en dépendant :

— R'_0 - $(\forall c, c_1, c_2 \neq ♠)$, |c| ≤ 5 et pas $(|c_1| = 5$ et $|c_2| = 5)$
— R'_1 - ≥ 13HCP et |♠| ≤ 1
— R'_2 - ≥ 16HCP et |♠| = 2 et |♥| ≥ 3 et |♦| ≥ 3 et |♣| ≥ 3
— R'_3 - ≥ 20HCP

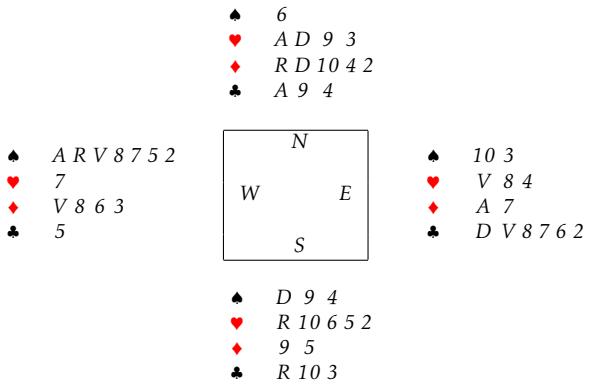
Après le même processus de mise au point avec les experts, la validation finale a demandé la génération de 70 000 000 donnes dont 10 007 vérifiaient au moins une règle d'enchère de 4♠ concernant la main d'Ouest et au moins une règle du Contre pour la main de Nord. Aucune règle relative au Contre n'a de support vide, R'_1 possède par exemple un support de 69.3% et R'_2 de 24.8%.

3 Génération automatique des données

3.1 Génération des donnes

1 000 donnes dont la main d'Ouest satifait au moins une règle de l'enchère de 4♠ et la main de Nord au moins une règle du Contre ont été générées. Le Passe du joueur Est n'est régi par aucune règle, ce qui est proche de la situation réelle.

Exemple 1 *La donne suivante (Nord-Sud vulnérables/Est-Ouest non vulnérables) a par exemple été générée et filtrée, la main d'Ouest vérifiant* (R_0, R_5) *et celle de Nord* (R'_0, R'_1) *:*



Pour chacune de ces 1 000 donnes, la main de Sud a été fixée et 999 donnes vérifiant les règles concernant les mains d'Ouest et de Nord ont été générées (temps de calcul moyen : 40", temps de calcul maximum : 10-15'). 1000 fichiers « exemples », contenant chacun 1 000 donnes possédant la même main en Sud, ont ainsi été créés.

Exemple 2 *À partir de cette donne, une des 999 autres donnes générées est la suivante. Main d'Ouest :* ♠*AR108653* ♥*4* ♦*832* ♣*84 - Main de Nord :* ♠*V2* ♥*AV73* ♦*AR106* ♣*AV9 Main d'Est :* ♠*7* ♥*D98* ♦*DV74* ♣*D7652 - Main de Sud :* ♠*D94* ♥*R10652* ♦*95* ♣*R103. La main de Sud est identique, la main d'Ouest vérifie cette fois* (R_0, R_2) *et celle de Nord* (R'_0, R'_2)*.*

3.2 Étiquettage des données

L'objectif est d'étiqueter chacune des 1 000 mains de Sud, donc chacun des 1 000 fichiers exemples, de la manière suivante :

— « **Pass** » quand la meilleure décision de Sud est de passer et que le camp adverse joue donc $4♠^X$.
— « **Bid** » quand la meilleure décision de Sud est au contraire d'enchérir et donc de jouer un contrat dans son camp.
— « **?** » quand la meilleure décision n'est pas possible à déterminer, afin d'exclure ces mains des données utilisées par les systèmes ILP.

Les données ne sont pas étiquetables directement par les experts dans la mesure où il suppose de calculer la meilleure décision en fonction d'une main de Sud donnée ce qui est difficile et coûteux. Nous avons donc procédé à un étiquetage automatique de la manière suivante :

Calcul des scores Pendant le jeu de la carte, chaque joueur voit sa propre main et l'une des mains du camp en attaque appelée le « mort ». Un solveur à quatre jeux (ou « DDS » pour « Double Dummy Solver ») permet de déterminer combien de levées seraient réalisées par un déclarant et des défenseurs voyant les quatre mains (Pavlicek, 2014). Connaissant le contrat, le score en est déduit de manière déterministe.

Sur les 1 000 donnes que comporte chaque fichier exemple ayant une main commune en Sud, un DDS a été utilisé pour déterminer les scores :

— de $4♠^X$ joué par le camp Est/Ouest
— de tous les autres contrats joués par le camp Nord/Sud à l'exception de certains, exclus a priori par les experts (4NT, 5NT, 5♠, 6♠ et 7♠)

Attribution des étiquettes Les étiquettes sont attribuées de la manière qui suit.

— « Pass » dans deux cas :
  — quand Sud marque en moyenne un meilleur score à $4♠^X$ qu'en enchérissant, même supposant qu'il découvrira par la suite le meilleur contrat dans son camp (meilleur atout, meilleur palier) ce qui n'est pas assuré en pratique.
  — quand Sud marque en moyenne un score significativement meilleur (30 points de plus) à $4♠^X$ qu'à tout autre contrat choisi définitivement.
— « Bid » quand il existe un contrat auquel Sud marque en moyenne un meilleur score qu'à $4♠^X$.
— « ? » dans les autres cas

Le jeu de données initial contenant 1 000 fichiers exemples est réduit à un jeu S de 960 exemples après élimination des 40 étiquettes « ? » et se répartissant avec un ratio d'étiquettes « Bid » de 35.2%.comme suit : :

— 338 étiquetés « Bid »
— 622 étiquetés « Pass »

3.3 Modélisation relationnelle

Nous présentons d'abord la théorie du domaine choisie destinée aux algorithmes d'apprentissage relationnel. La plupart des prédicats utilisés ont été introduits dans (Legras et al., 2018), parmi lesquels $gteq(A,B)$ ($A \geq B$), $lteq(A,B)$ ($A \leq B$) ou encore $nb(Hand, Suit, Number)$ (la couleur $Suit$ d'une main $Hand$ a pour longueur $Number$). Par exemple, $nb(Hand, heart, 3)$ signifie que la main Hand contient 3 cartes à ♥.

Prédicat cible Étant donné une main, une position, la vulnérabilité ainsi qu'un donneur, le but est de prédire l'étiquette Class.

Le prédicat cible est $decision(Hand, Position, Vul, Dealer, Class)$ avec :

— $Hand$: les 13 cartes du joueur devant prendre la décision « Bid » ou « Pass »
— $Position$: la position relative de ce joueur au donneur (ici toujours 4)
— Vul : la vulnérabilité (b = both, o = none, n = north-south ou e = east-west)
— $Dealer$: la position cardinale du donneur ($north$, $east$, $south$ ou $west$)
— $Class$: l'étiquette à prédire (bid ou $pass$)

Nouveaux prédicats Afin d'apprendre un modèle général et explicable, nous avons construit des prédicats abstraits permettant de saturer et de faire des inférences. Les deux prédicats de plus haut niveau non utilisés dans (Legras et al., 2018) sont :

— $hcp(Hand, Number)$: nombre de HCP de la liste de cartes $Hand$.

— *suit_representation*(*Hand*, *Suit*, *Honors*, *Number*) : abstraction d'une couleur d'une main. *Honors* est la liste des cartes supérieures au 10 de la couleur *Suit* dans la main *Hand*. *Number* est le nombre total de cartes de la couleur *Suit*.

Représentation d'une main Les bridgeurs abstraient les distributions en une partition de trois classes : les mains régulières ou « *balanced* » (sans couleur de 0 ou 1 carte, et au plus une couleur de 2 cartes), les mains semi-régulières ou « *semi_balanced* » (sans couleur de 0 ou 1 cartes, exactement deux couleurs de 2 cartes) et les mains irrégulières ou « *unbalanced* ».

On peut ainsi changer à loisir la granularité du langage d'hypothèse en choisissant de donner la distribution exacte, la distribution ou les prédicats *balanced*, *semi_balanced* et *unbalanced*.

4 Apprentissage de règles expertes à l'aide de systèmes de PLI

4.1 Les systèmes choisis

Les systèmes de PLI (Muggleton et Raedt, 1994) que nous avons utilisés dans nos expériences sont Aleph (Srinivasan, 1999) et Tilde (Blockeel et al., 1999). Ce sont deux systèmes de PLI de l'état de l'art, assez différents dans leur stratégie d'apprentissage.

Aleph apprend par implication (*learning from entailment*) (Muggleton et Raedt, 1994). Étant donné un ensemble d'exemples positifs E^+, un ensemble d'exemples négatifs E^- et une théorie du domaine T, le but de l'apprentissage est de trouver un programme logique (un ensemble de clauses) H, tel que $\forall e \in E^+ : H \wedge T \models e$ et $\forall e \in E^- : H \wedge T \not\models e$. Pour Aleph, les exemples sont des littéraux représentant la classe ($bid/1$). La théorie du domaine est représentée par un ensemble de faits et de règles qui vont permettre d'enrichir la représentation des exemples (voir section 3.3). Une partie de cette théorie du domaine a déjà été utilisée dans une autre application de ces mêmes systèmes de PLI pour la prise de décision dans un problème d'enchères (Legras et al., 2018), ce qui illustre bien la réutilisabilité de cette théorie du domaine. Aleph dans sa version standard (`induce`) apprend des règles en utilisant une stratégie de couverture : il sélectionne un exemple graine e non couvert par la théorie courante. Il parcourt ensuite de façon descendante un espace de recherche dont la borne inférieure est une clause maximalement spécifique (notée B) construite à partir de la graine e et de la théorie du domaine. Une fois une règle R_e apprise pour e, les exemples positifs couverts par R_e sont effacés de l'ensemble d'apprentissage et l'apprentissage reprend sur ce nouvel ensemble, jusqu'à ce que tous les exemples positifs soient couverts. A cause du choix de la graine e, Aleph est sensible à l'ordre des exemples d'apprentissage lorsque la stratégie de parcours par défaut (`best_first`) est utilisée. Nous avons également utilisé Aleph avec une stratégie plus coûteuse (`induce_max`) insensible à l'ordre des exemples positifs. Elle construit une unique couverture de l'ensemble des exemples positifs en choisissant la meilleure clause (en terme de couverture) couvrant chaque exemple positif. Les clauses obtenues ont alors un recouvrement important, ce qui s'est avéré bénéfique dans nos expériences.

Tilde apprend un arbre de décision relationnel c'est-à-dire un arbre de décision binaire dans lequel les sommets de l'arbre sont des conjonctions de littéraux pouvant partager des variables, avec la restriction suivante : une variable introduite dans un sommet ne peut pas apparaître dans la branche droite sous ce sommet (la branche d'échec du test porté par le sommet). Afin de spécialiser un sommet de requête associée Q, Tilde calcule une spécialisation de Q étant donné le biais de langage défini par l'utilisateur et choisit la spécialisation qui maximise un critère de qualité, par défaut le gain d'information adapté au cas relationnel.

4.2 Paramétrage d'Aleph et Tilde

Nous abordons ici les différents choix de paramètres retenus pour Aleph et Tilde pour l'expérience présentée dans la section suivante. Certains de ces choix sont inhérents à l'algorithme utilisé, d'autres ont été réalisés pour des raisons de performances, de lisibilité du modèle induit ou par souci de simplicité d'implémentation. Nous ne présentons que les paramétrages les plus significatifs pour chaque système (Srinivasan, 1999; Blockeel et al.).

Biais de langage Le biais de langage spécifie pour les deux algorithmes de PLI l'espace de recherche, en décrivant quels prédicats vont former les hypothèses, et quels arguments peuvent prendre ces prédicats : pour chaque prédicat d'arité non nulle, on peut préciser la nature de ses arguments (variable ou constante, entrée ou sortie [1]). De plus, Aleph peut manipuler des termes complexes. Il peut ainsi au cours de sa recherche construire une clause dont un des termes est une liste partiellement instanciée : $distribution(A, [B, C, 2, 1])$ dénote une distribution d'une main dont les deux couleurs les plus courtes sont 2 et 1, les longueurs des deux couleurs les plus longues étant non spécifiées. La fonctionnalité `lookahead` de Tilde lui permet de créer des noeuds testant une conjonction de littéraux. L'usage de $nb(E, Suit, Value)$, et de $gteq$ ou $lteq$ lui permet de décrire la longueur d'une couleur spécifique ou non et d'ainsi généraliser la notion de distribution en effectuant de multiple tests sur les différentes couleurs.

Certains biais de langage ont aussi été introduits en concertation avec les experts du domaine, après des expériences d'apprentissage préliminaires. Par exemple, ils n'étaient pas satisfaits de certains prédicats apparaissant dans les hypothèses trop spécifiques autour d'une notion qu'ils estimaient présentant une symétrie. Après rectification, les modèles construits ne faisaient plus intervenir cette notion, sans perte de précision sur les ensembles de tests, validant *a posteriori* le biais de langage choisi.

Options de recherche Nous avons choisi les options de recherche des algorithmes par défaut quand cela était possible pour Aleph et Tilde. Les paramètres de Tilde sont exclusivement ceux par défaut. Pour Aleph : *clauselength*=6 (la longueur maximum de la règle générée), *minpos*=5 (le nombre minimum d'exemples positifs couverts par la régle) , *minacc* = 0.85 et *noise*= inf (le nombre maximum d'exemples négatifs couverts

1. Une variable en entrée apparaît dans la partie gauche de la clause en construction, alors qu'une variable de sortie n'apparaît pas encore dans la clause courante et est donc existentiellement quantifiée.

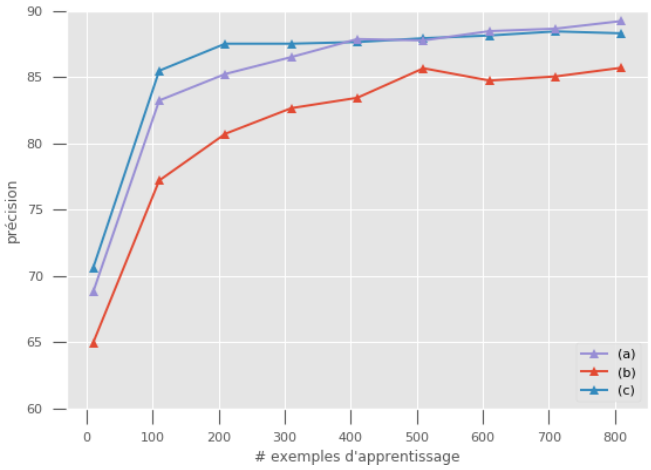


FIG. 1 – *Précision moyenne de Tilde (a), Aleph* `induce` *(b) et Aleph induce_max (c)*

par une règle) ; pour Tilde : *heuristic* = gainratio, *pruning* = c45 safe pruning, *stopping criterion* = mincase.

5 Protocole expérimental

La performance du système d'apprentissage vis-à-vis de la taille de l'ensemble d'apprentissage est une moyenne sur 50 exécutions i. Les ensembles $Test_i$ (test) et $Train_i$(apprentissage) sont constitués comme suit :

— $|Test_i| = 0.1 \times |S| = 96$

— Chaque $Test_i$ est stratifié, i.e. il comporte le même ratio d'exemples étiquetés bid que S (35.2%)

— $Train_i = S \setminus Test_i$

Pour chaque exécution i et une séquence de nombre d'exemples croissante $n_k = 10 + 100 \times k$ avec k entre 0 et 8, on tire dans $Train_i$ des sous ensembles $T_{i,k}$ de taille n_k tel que $T_{i,k} \subset T_{i,k+1}$. De plus, à chaque construction d'un ensemble $T_{i,kf}$, ses éléments sont réordonnés aléatoirement (Aleph est sensible à l'ordre des exemples).

Chaque modèle appris sur $T_{i,k}$ est évalué sur $Test_i$. La Fig.1 présente la précision moyenne sur $i = 1 \ldots 50$ exécutions des modèles appris sur les ensembles $T_{i,k}$ en fonction de n_k avec Tilde et Aleph (`induce` et `induce_max`).

De premières expérimentations avec Tilde et Aleph `induce` donnent dans un premier temps un net avantage de précision aux modèles appris par Tilde (Fig.1, courbe (a)) par rapport aux modèles appris par Aleph (induce) (Fig.1 courbe (b)). Après étude et post-traitement d'un de ces modèles, nous avons cependant pu construire manuellement un programme logique ayant une bonne précision qu'Aleph aurait pu

découvrir, modulo quelques modifications dans sa configuration. Nous avons donc modifié le biais de langage et la stratégie de recherche d'Aleph afin d'apprendre des modèles (courbe (c) sur Fig.1) jugés satisfaisant par les experts du domaine et de performances similaires à Tilde. Nous décrivons ce processus dans la section suivante.

6 Analyse experte des règles, retour sur l'apprentissage

Au-delà des performances en apprentissage, mesurées ici par la précision sur l'ensemble de test, nous avons soumis les modèles jugés les plus performants à l'analyse des experts du domaine. La sélection d'un modèle Aleph et d'un modèle Tilde a été effectuée comme suit. Parmi tous les modèles appris, nous avons sélectionné les 5 meilleurs modèles en précision sur $Test_i$ (voir section 5). En cas d'égalité de performance, nous avons choisi les modèles appris sur le plus petit ensemble d'apprentissage, et enfin en cas d'égalité, les modèles les plus simples (nombre de règles pour Aleph ou nombre de sommets de l'arbre pour Tilde).

Listing 1 – *Extrait du meilleur arbre appris par Tilde*

```
decision(-A,-B,-C,-D,-E)
nb(A,-F,-G),gteq(G,6) ?
+--yes: hcp(A,4) ?
|       +--yes: [pass]
|       +--no:  [bid]
+--no:  hcp(A,-H),gteq(H,16) ?
        +--yes: [bid]
        +--no:  nb(A,spade,-I),gteq(I,1) ?
                +--yes: lteq(I,1) ?
                |       +--yes: hcp(A,-J),lteq(J,5) ?
                |       |       +--yes: [pass]
                |       |       +--no:  nb(A,-K,5) ?
                |       |               +--yes: hcp(A,6) ?
                |       |               |       +--yes: [pass]
                |       |               |       +--no:  [bid]
                |       |               +--no:  [pass]
```

L'arbre de Tilde dont un extrait est donné Listing 1 est concis (15 noeuds) mais reste difficilement interprétable. Sa traduction sous forme de liste de décision ne résolvant pas ce problème, nous avons procédé à une extraction manuelle et experte de règles issues de l'arbre de la façon suivante :

1. Traduction brute en langage naturel, traitement des négations d'intervalles.
2. Extraction des règles qui concluent sur une feuille *bid*.
3. Suppression de noeuds non pertinents à support faible (en particulier, tests sur les valeurs précises de points HCP)
4. Reformulation des intervalles (points HCP, nombre de cartes).
5. Mise en avant du nombre de cartes à ♠ et des distributions génériques.

Les règles obtenues (voir Listing 2) sont bien plus lisibles que l'arbre de Tilde pour une précision sur l'ensemble de test légèrement dégradée : 90.62% au lieu de 96.91%.

Listing 2 – *Règles extraites du meilleur arbre de Tilde*

```
On encherit :
– Quel que soit le nombre de ♠, avec une couleur 6e+ ou 16+HCP
– Avec 0♠
– Avec 1♠ si 5–4–3–1♠ ou 5–5–2–1♠ dans l'intervalle 6–15HCP
– Avec 2♠ si 5–5–2♠–1 dans l'intervalle 7–15HCP
– Avec 2♠ si 5–4–2–2♠ dans l'intervalle 11–15HCP
```

Nous avons relancé un apprentissage Aleph avec le langage d'hypothèses minimum permettant de construire ces règles. Les modèles ainsi obtenus ont une meilleure précision par rapport à la première version d'Aleph, mais encore en deçà de la performance de Tilde. De plus, les règles étaient encore trop nombreuses et spécifiques relativement aux règles extraites manuellement.

Il a alors été décidé d'utiliser la stratégie `induce_max` d'Aleph au lieu de `induce`. La recherche `induce_max` a un coût plus élevé en temps et les modèles obtenus présentent des hypothèses redondantes. La précision moyenne des modèles ainsi appris par Aleph `induce_max` avec un langage minimaliste est montrée à la Fig.1 (c). La courbe de Tilde est bien dominée cette fois par celle d'Aleph. Voici ci-dessous les règles apprises par Aleph avec `induce_max` (Listing 3), où le prédicat *nbs* dénote le nombre de cartes à ♠ dans la main du joueur.

Listing 3 – *Meilleur modèle appris par Aleph*

```
decision(A,4,B,C,bid) :- nb(A,D,E), gteq(E,6).
decision(A,4,B,C,bid) :- hcp(A,D), gteq(D,14), nb(A,E,F), lteq(F,2).
decision(A,4,B,C,bid) :- hcp(A,D), lteq(D,11), gteq(D,11),
                                    distribution(A,[5,4,2,2]).
decision(A,4,B,C,bid) :- nbs(A,0).
decision(A,4,B,C,bid) :- hcp(A,D), gteq(D,9), nbs(A,1).
decision(A,4,B,C,bid) :- hcp(A,D), gteq(D,7), nb(A,E,F), gteq(F,5), nbs(A,1).
decision(A,4,B,C,bid) :- hcp(A,D), gteq(D,13), nb(A,E,F), gteq(F,5), nbs(A,2).
decision(A,4,B,C,bid) :- nb(A,D,E), lteq(E,1), nbs(A,2).
```

Le meilleur modèle d'Aleph (induce_max) a été appris sur un ensemble d'apprentissage de 410 exemples, sa précision est de 97.94%. Celui de Tilde (Listing 1) a été obtenu avec un ensemble d'apprentissage de taille 310 et a une précision de 96.91%.

7 Conclusion

Nous avons pu mener de bout en bout une expérience d'acquisition de règles expertes pour un problème d'enchères au bridge. Les résultats obtenus valident nos hypothèses de travail. La méthodologie de construction des exemples recourt à la fois aux experts (modélisation et représentation relationnelle des exemples) et à une boîte noire performante pour l'étiquetage. Cette méthodologie permet de produire un ensemble d'exemples de qualité, qui est un préalable pour l'apprentissage d'un modèle relationnel compact et de très haute performance. Ce modèle relationnel final a été construit après une phase supplémentaire d'interactions avec les experts, qui a permis de préciser encore le langage de concepts. Cette interaction a été rendue possible car les modèles relationnels appris sont interprétables. Nous explorerons dans de futurs travaux des méthodes de transformation automatique d'arbres de décision relationnels en ensembles de règles.

Références

ACBL (2019). How to play Bridge. https://www.acbl.org/learn_page/how-to-play-bridge/.

Blockeel, H., L. Dehaspe, J. Ramon, J. Struyf, A. V. Assche, C. Vens, et D. Fierens. The ACE data mining system user's manual. Available at https://dtai.cs.kuleuven.be/ACE/doc/ACEuser-1.2.16.pdf.

Blockeel, H., L. D. Raedt, N. Jacobs, et B. Demoen (1999). Scaling up inductive logic programming by learning from interpretations. *Data Min. Knowl. Discov. 3*(1), 59–93.

Féderation Française de Bridge (2018). Je découvre. https://www.ffbridge.fr/j-apprends#je-decouvre.

Legras, S., C. Rouveirol, et V. Ventos (2018). The game of bridge : a challenge for ilp. In *Inductive Logic Programming 28th International Conference, ILP 2018 Proceedings*, pp. 72–87. Springer.

Muggleton, S. et L. D. Raedt (1994). Inductive logic programming : Theory and methods. *J. Log. Program. 19/20*, 629–679.

Murdoch, W. J., C. Singh, K. Kumbier, R. Abbasi-Asl, et B. Yu (2019). Interpretable machine learning : definitions, methods, and applications. *arXiv e-prints*, arXiv :1901.04592.

Patterson, D. (2008). Computer learning in Bridge. Msci Individual Project, Queen Mary College, University of London.

Pavlicek, R. (2014). Actual play vs. double-dummy. Available at http://www.rpbridge.net/8j45.htm.

Ribeiro, M. T., S. Singh, et C. Guestrin (2016). "Why Should I Trust You?" : Explaining the predictions of any classifier. In *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144.

Srinivasan, A. (1999). The Aleph manual. Available at http://www.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/.

Université du Bridge (2013). *Le Bridge Français*, Volume 1. Pole edition.

Summary

We address the problem of building a decision model for a specific bidding phase in the game of Bridge. We propose the following multi-step methodology. We first build a representative example set for the decision problem at hand and use simulations with a black-box solver to associate a bidding decision to each example. Then, supervised relational learning builds an explicit interpretable model fitting the black-box model as well as possible. Interactions between game and machine learning experts that evaluate and gradually improve the successive models produced make this methodology successful.