Une méthode d'apprentissage par optimisation multicritère pour le rangement de motifs en fouille de données

Nassim Belmecheri * **, Noureddine Aribi*, Nadjib Lazaar**, Yahia Lebbah*, Samir Loudni***

*Lab. LITIO, Université Oran1, 31000 Oran, Algérie
belmecheri.nassim@edu.univ-oran1.dz,{ylebbah, aribi.noureddine}@gmail.com
**LIRMM, Université de Montpellier, CNRS, Montpellier, France
Nadjib.Lazaar@lirmm.fr
***TASC (LS2N-CNRS), IMT Atlantique, FR – 44307 Nantes, France
samir.loudni@imt-atlantique.fr

Résumé. La découverte de motifs pertinents est une tâche difficile en fouille de données. D'une part, des approches ont été proposées pour apprendre automatiquement des fonctions de rangement de motifs spécifiques à l'utilisateur. Ces approches sont souvent efficaces en qualité, mais très coûteuses en temps d'exécution. D'autre part, de nombreuses mesures d'intérêt sont utilisées pour évaluer l'intérêt des motifs dans le but de se rapprocher le plus possible du rangement de l'utilisateur. Dans cet article, nous formulons le problème d'apprentissage des fonctions de rangement des motifs comme un problème d'optimisation multicritère. L'approche proposée permet d'agréger des mesures d'intérêt en une fonction linéaire pondérée dont les poids sont calculés via la méthode AHP (Analytic Hierarchy Process). Des expérimentations menées sur de nombreux jeux de données montrent que notre approche réduit drastiquement le temps d'exécution, tout en assurant un rangement comparable à celui des approches existantes.

1 Introduction

La découverte des motifs est l'un des problèmes fondamentaux en fouille de données ensemblistes. En raison du grand nombre de motifs découverts dans un ensemble de données, les derniers travaux s'intéressent de plus en plus à l'extraction des motifs les plus pertinents. Cependant, l'utilisation des méthodes classiques d'extraction de motifs pose de nombreuses difficultés en raison de la combinatoire complexe pour garantir à la fois la qualité des motifs et un temps d'exécution acceptable. Les deux principaux problèmes sont : 1) des quantités importantes de motifs sont découverts, avec beaucoup de redondance ; 2) les préférences d'un expert du domaine ne sont pas prises en compte. L'importance de prendre en compte les préférences des utilisateurs a été discutée pour la première fois par Silberschatz et Tuzhilin (1995). L'idée de base est de modéliser les hypothèses des utilisateurs à l'aide de mesures d'intérêt qui se basent sur la structure des données. Toutefois, comme indiqué dans Bie (2011), l'utilisation de mesures d'intérêt est limitée lorsqu'il s'agit de capturer les préférences de l'utilisateur.

Récemment, l'extraction des motifs a été mise en défi avec la fouille interactive orientée utilisateur Dzyuba et van Leeuwen (2013); Boley et al. (2013); Dzyuba et van Leeuwen (2017). Ce nouveau paradigme met l'accent sur la contribution de l'utilisateur dans le processus d'extraction de motifs dès le début en lui présentant des motifs susceptibles de l'intéresser et ainsi conditionner les itérations suivantes en prenant en compte ses retours, ce qui permet d'apprendre des fonctions de rangement des motifs spécifiques à l'utilisateur. Cette idée a été étudiée pour la première fois dans Xin et al. (2006) et récemment étendue par Boley et al. (2013) et Dzyuba et van Leeuwen (2013, 2017). Même si ces méthodes exploitent efficacement les fonctions de rangement, elles souffrent d'un temps d'exécution trop coûteux, notamment avec l'augmentation des retours de l'utilisateur. Dans ce papier, nous nous intéressons à la problématique d'apprentissage des fonctions de rangement en exploitant l'optimisation multicritère et l'agrégation des mesures individuelles en une somme pondérée. Les bonnes performances de notre approche consistent en sa capacité à calculer rapidement les coefficients de la somme pondérée tout en garantissant une qualité aussi bonne que celle des méthodes de la littérature. L'approche proposée exploite la méthode AHP ("Analytical Hierarchy Process"), qui permet de déduire les poids de la somme pondérée à partir des rangements fournis par l'utilisateur, en maximisant la corrélation avec les rangements fournis par l'utilisateur. Pour montrer l'intérêt de notre approche, nous avons exploité un cas d'étude sur la fouille des règles d'association. Les expérimentations menées sur des ensembles de données variés, montrent que notre approche réduit drastiquement le temps d'exécution, tout en garantissant une bonne qualité d'apprentissage par rapport aux méthodes de la littérature.

2 Préliminaires

Un ensemble de données peut être abstrait sous forme d'un triplet $(\mathcal{A}, \mathcal{O}, \mathcal{L})$, où $\mathcal{A}$ est un ensemble d'attributs, $\mathcal{O}$ est un ensemble d'objets et $\mathcal{L}$ est le langage utilisé pour exprimer des objets sur des attributs. Ce cadre est inspiré des concepts des bases de données inductives Raedt (2002); Imielinski et Mannila (1996). Concernant le langage $\mathcal{L}$, nous citons deux exemples : (1) si $\mathcal{L}$ est une relation binaire entre les attributs et les objets (i.e., $\mathcal{L} \subseteq \mathcal{O} \times \mathcal{A}$), nous obtenons le cadre de la fouille de données ensemblistes ; (2) si $\mathcal{L}$ exprime chaque objet comme une séquence de présence d'attributs, nous obtenons une base de données séquentielle. Dans ce papier nous nous intéressons à la fouille de motifs ensemblistes.

Mesures d'intérêt des motifs : Pour qualifier le degré d'intérêt d'un motif dans un ensemble de données, de nombreuses mesures $M_i : \mathcal{L} \rightarrow \mathbb{R}$ ont été proposées dans le contexte des méthodes développées pour la découverte de motifs (voir Geng et Hamilton (2006)). Étant donné une mesure d'intérêt M_i, nous pouvons définir une relation de préférence binaire sur $\mathcal{L}$ comme suit : $P_1 \rhd_{M_i} P_2 \Leftrightarrow M_i(P_1) \geq M_i(P_2)$, ce qui signifie que le motif P_1 est préféré au motif P_2. Cette relation de préférence peut être étendue à un modèle de préférences subjectives binaire, tel que : le motif P_1 est préféré au motif P_2 (noté $P_1 \rhd_u P_2$) si et seulement si $P_1 \succsim P_2$, où $\succsim$ est un pré-ordre total.

Agrégation des mesures : Identifier les motifs qui sont les plus intéressants pour un utilisateur est une tâche difficile. Des approches de l'état de l'art combinent des mesures d'intérêt afin de produire une fonction but pour générer les meilleurs motifs. Ceci peut être fait en associant des poids respectifs aux différentes mesures, et en les agrégeant selon des opérateurs d'agrégation

connus Figueira et al. (2005). Dans ce papier, nous considérons une *fonction d'agrégation linéaire* g pour agréger des mesures individuelles.

Définition 1 (Fonction d'agrégation linéaire) *Étant donné un motif P, un ensemble de mesures $\mathcal{M} = \{M_1, \ldots, M_m\}$ et un vecteur de poids $w = \langle w_1, \ldots, w_m \rangle$, avec $w_i \in \mathbb{R}$, une fonction pondérée d'agrégation linéaire g_w est définie comme suit : $g_w(P) = \sum_{i=1..m} w_i.M_i(P)$.*

La méthode AHP : AHP (The Analytical Hierarchy Process) est une approche simple de décision multicritère Saaty (1988). Elle est utilisée pour organiser et éliciter les poids d'une fonction d'agrégation linéaire. Elle consiste en deux étapes : (1) Les critères sont subjectivement comparés paire par paire, relativement aux préférences subjectives de l'utilisateur. Cette comparaison est organisée sous forme d'une matrice carrée $A = [1..m, 1..m]$, où $A[i, j]$ est l'importance relative d'un critère i par rapport à un autre critère j. Le $i^{\text{ème}}$ critère est meilleur que le $j^{\text{ème}}$ critère si $(A[i, j] > 1)$. Dans AHP, nous disposons de 9 degrés de préférence où $A[i, j] = 1$ indique une indifférence et $A[i, j] = 9$ une préférence absolue. Notons que $A[i, j] = 1/A[j, i]$ et $A[i, i] = 1$. (2) A partir de la matrice A, AHP calcule les poids des critères $w \in \mathbb{R}^m$ avec une méthode d'optimisation (voir Brunelli (2015)).

3 Problématique de rangement

Soient $\triangleright$ une relation binaire de préférence et $\mathcal{P} = \{P_1, \ldots, P_k\}$ un ensemble de motifs. Nous introduisons la fonction de rangement $r_\triangleright$, qui ordonnera totalement les motifs dans $\mathcal{P}$. Plus précisément, $r_\triangleright(\mathcal{P}) = \langle P_{r_1}, P_{r_2}, \ldots, P_{r_k} \rangle$ est une permutation de $(P_1, P_2, \ldots, P_k)$. Nous dénotons par $r_\triangleright(\mathcal{P}, P_i)$ le rang du motif P_i dans $\mathcal{P}$ relativement à la relation de préférence $r_\triangleright$. Soit un ensemble de k motifs $\mathcal{P} = \{P_1, ..., P_k\}$, pour comparer deux rangements $r_{\triangleright_x}(\mathcal{P})$ et $r_{\triangleright_y}(\mathcal{P})$, nous utilisons le coefficient de corrélation de Kendall et Smith (1939) $K_{x,y}$ pour évaluer de combien deux rangements sont proches : $K_{x,y} = \frac{3SM}{(|\mathcal{P}|^3 - |\mathcal{P}|)}$ où $SM = \sum_{i=1}^{|\mathcal{P}|}(R_i - \bar{R})^2$, $R_i = r_{\triangleright_x}(\mathcal{P}, P_i) + r_{\triangleright_y}(\mathcal{P}, P_i)$ et $\bar{R} = \frac{1}{|\mathcal{P}|}\sum_{i=1}^{|\mathcal{P}|} R_i$. Soit un ensemble de motifs $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$. Nous présentons cet ensemble à l'utilisateur pour ordonner complètement les motifs en se basant sur son jugement subjectif. Soit $S = \langle P_{r_1}, \ldots, P_{r_k} \rangle$ une liste de motifs ordonnés par l'utilisateur, où $\forall i < j$, $P_{r_i} \triangleright_u P_{r_j}$. Notre objectif est d'exploiter le rangement de l'utilisateur $\mathcal{S} = \{S_1, \ldots, S_n\}$ afin d'apprendre une fonction de rangement $r_{\triangleright_u}$.

Définition 2 (Problème d'apprentissage de la fonction de rangement de l'utilisateur) *Soient un ensemble de données ($\mathcal{A}$, $\mathcal{O}$, $\mathcal{L}$), un ensemble de mesures $\mathcal{M} = \{M_1, ..., M_m\}$, un ensemble de motifs $\mathcal{P} = \{P_1, ..., P_k\}$ de cardinalité k, et le rangement de l'utilisateur $\mathcal{S} = \{S_1, \ldots, S_n\}$. Il s'agit d'apprendre les poids w induits par $\mathcal{S}$ qui maximisent la corrélation $K_{g_w,u}$ entre les fonctions de rangement $r_{\triangleright_{g_w}}$ et $r_{\triangleright_u}$.*

4 La méthode AHP pour le rangement des motifs

Algorithme 1 : `AHPRank`

Input : $\mathcal{M} = \{M_1, \ldots, M_m\}$: m mesures ; $\mathcal{S} = \{S_1, \ldots, S_n\}$: ensemble de motifs rangés par l'utilisateur ;
Output : w : vecteur de poids ;

$\Delta = \{\Delta_{i,j} : i, j \in 1 \ldots m \wedge i < j\}$;
foreach $\Delta_{i,j} \in \Delta$ **do** $\Delta_{i,j} \leftarrow 0$;
$\Delta \leftarrow$ `calculerCorrelation`$(\Delta, \mathcal{S}, \mathcal{M})$;
$A \leftarrow$ `créerMatriceAHP`(Δ) ;
return `calculerPoids`(A) ;

Function `calculerCorrelation`$(\Delta, \mathcal{S}, \mathcal{M})$ **:**
  $l \leftarrow 0$;
  foreach $S_k \in \mathcal{S}$ **do**
    foreach $M_i \in \mathcal{M}$ **do** Calculer $K_{i,k}$;
    foreach $M_i, M_j \in \mathcal{M} : i < j$ **do** $\Delta_{i,j} \leftarrow \frac{l}{l+1}\Delta_{i,j} + \frac{1}{l+1}(K_{i,k} - K_{j,k})$;
    $l \leftarrow l + 1$;
  end
  return Δ ;

Function `créerMatriceAHP`(Δ) **:**
  $A[i,j] \leftarrow 1, \forall i, j \in \{1, \ldots, m\}$;
  foreach $\Delta_{i,j} \in \Delta$ **do**
    if $\Delta_{i,j} < 0$ **then** scale $\Delta_{i,j}$ to $(-9..-1)$; $A[j,i] \leftarrow |\Delta_{i,j}|$; $A[i,j] \leftarrow 1/|\Delta_{i,j}|$;
    else scale $\Delta_{i,j}$ to $(1..9)$; $A[i,j] \leftarrow \Delta_{i,j}$; $A[j,i] \leftarrow 1/\Delta_{i,j}$;
  end
  return A ;

Soit $\mathcal{S} = \{S_1, \ldots, S_n\}$ un ensemble de motifs rangés par un utilisateur, avec $S_i = \langle P_{i_1}, \ldots, P_{i_k} \rangle$. Soit $\mathcal{M} = \{M_1, \ldots, M_m\}$ un ensemble de mesures d'intérêt. Dans cette section, nous présentons `AHPRank`, une approche d'agrégation linéaire des mesures d'intérêt afin de produire une mesure agrégée efficace pour la découverte des motifs pertinents. Ceci est réalisé grâce à la technique AHP. AHP nécessite une matrice carrée $A[1..m, 1..m]$, où $A[i,j]$ indique l'intensité avec laquelle le critère i est préféré par rapport au critère j.

Mesure d'intérêt : Pour évaluer la précision globale du rangement d'une mesure donnée par rapport au rangement de l'utilisateur, nous utilisons le coefficient de concordance W de Kendall et Smith (1939). Étant donné une mesure M_i et des motifs rangés par l'utilisateur S_j, nous désignons par $K_{i,j} \in [0,1]$, le W de Kendall entre le rangement des motifs utilisant M_i et les motifs rangés par l'utilisateur S_j, où la valeur 1 (resp., la valeur 0) représente une concordance parfaite (resp., aucune concordance).

Comparer deux mesures : Soient M_i et M_j deux mesures à comparer sur un ensemble de motifs rangé par l'utilisateur S_k. $\Delta_{i,j,k}$ est une différence entre la valeur K des deux mesures : $\Delta_{i,j,k} = (K_{i,k} - K_{j,k})$, si $\Delta_{i,j,k} \geq 0$, alors M_i est meilleure que M_j, sinon l'inverse. Enfin, pour comparer deux mesures M_i et M_j sur l'ensemble des données d'apprentissage $\mathcal{S}$, nous pouvons estimer la mesure la plus intéressante en faisant la moyenne de leurs écarts sur l'ensemble des motifs rangés par l'utilisateur S_k : $\Delta_{i,j} = \frac{1}{n} \sum_{S_k \in \mathcal{S}} \Delta_{i,j,k}$.

Algorithme AHPRank : L'algorithme 1 calcule un vecteur de poids w (un poids pour chaque mesure) sans rangement préalable des mesures, en exploitant les comparaisons de paires de mesures. L'algorithme 1 prend en entrée un ensemble de mesures $\mathcal{M}$ et un ensemble de motifs rangés par l'utilisateur $\mathcal{S}$. L'algorithme 1 commence par initialiser le vecteur de comparaison des paires Δ à zéro (ligne 2). L'étape d'apprentissage est effectuée à la ligne 3, où pour chaque paire de mesures, nous calculons l'écart moyen sur tous les motifs rangés par l'utilisateur S_k (fonction `calculerCorrelation` à la ligne 6). Ensuite, l'algorithme 1 construit la matrice AHP A en mettant à l'échelle les comparaisons $\Delta_{i,j}$ aux valeurs autorisées de la matrice AHP A (fonction `créerMatriceAHP` à la ligne 14). Enfin, l'algorithme fait appel à la fonction `calculerPoids` en exploitant une méthode d'optimisation basée sur le calcul du vecteur propre associé à la plus grande valeur propre de la matrice A (voir Brunelli (2015)). On peut démontrer que la complexité de l'algorithme 1 est de $\mathcal{O}(n\ k)$, où n est le nombre de requêtes et k est la taille de la plus grande requête.

L'agrégation AHP comme mesure d'intérêt : Une fois le vecteur de poids w calculé, nous pouvons calculer le score d'un motif donné P en utilisant la fonction d'agrégation linéaire suivante : $g_w(P) = \sum_{M_i \in \mathcal{M}} w_i\ scale(M_i(P))$. où la fonction $scale$ est utilisée pour ajuster les valeurs d'intérêt des différentes mesures à une échelle commune dans l'intervalle $[0, 1]$.

Version passive de AHPRank : L'algorithme 1 permet d'apprendre le vecteur de poids w à partir d'un ensemble de motifs rangés par l'utilisateur $\mathcal{S}$. Cet ensemble peut-être considéré comme une donnée statique et sera exploité comme une donnée d'apprentissage. Dans ce contexte, `AHPRank` agira en mode passif.

Version active de AHPRank : L'algorithme 1 peut être exploité en mode actif. Nous avons proposé une méthode d'apprentissage qui prend en compte l'apprentissage précédent en considérant Δ comme un paramètre d'entrée/sortie. Ici, Δ peut être calculé de manière incrémentale si les motifs rangés par l'utilisateur $\mathcal{S}$ évoluent dans le temps.

5 Expérimentations

Dans cette section, nous procédons à une étude expérimentale de notre méthode d'apprentissage pour le rangement des motifs. Notre étude de cas porte sur l'extraction de règles d'association, avec une description des mesures d'intérêt et du rangement de l'utilisateur.

Cas d'étude : Extraction des règles d'association : Nous évaluons notre approche générique sur la fouille des règles d'association Agrawal et al. (1993).

Tan et al. (2004) ont réalisé une étude intéressante sur l'utilité des mesures existantes en fonction du type d'application. Ils ont notamment identifié sept groupes indépendants de mesures cohérentes ayant des propriétés similaires. Nous avons sélectionné 7 mesures, une mesure par groupe, à savoir : Yules Q, Cosine, Laplace, ϕ-coefficient, Goodman-kruskal's, Intrest-factor et Certainty-factor.

Simulation de l'utilisateur : Nous simulons le rangement de l'utilisateur avec une mesure de qualité dédiée aux règles d'association, qui est considérée comme une mesure cible, non connue par l'algorithme d'apprentissage. Comme suggéré dans Dzyuba et van Leeuwen (2017), et selon Ringuest (1986), χ^2 est une mesure statistique appropriée pour la simulation de l'utilisateur. Le χ^2 est une fonction assez complexe à approximer. Étant donné une règle d'associa-

tion $A \Rightarrow B$, sa mesure associée χ^2 est définie : $\chi^2(A \Rightarrow B) = \frac{(freq(A,B) - \frac{freq(A)freq(B)}{N})^2}{\frac{freq(A)freq(B)}{N}}$, avec $freq(X)$ la fréquence de l'ensemble des éléments X et N le nombre de transactions.

Protocole expérimental : Nous avons implémenté en Java nos versions de `AHPRank` : la version passive désignée par `AHPRank.0` et la version active désignée par `AHPRank.1`. Nous avons comparé notre approche à l'état de l'art, l'approche basée sur SVM `RankingSVM` Dzyuba et van Leeuwen (2013). Nous désignons par `RankingSVM.0` la version passive et par `RankingSVM.1` la version active. Toutes les expériences ont été menées sur un Intel core $i7$, $2.4Ghz$ avec $16Gb$ de RAM en utilisant un timeout d'une heure. Nous considérons trois métriques pour évaluer les performances de notre approche : **(i)** le coefficient de corrélation des rangs de Spearman ρ, $\rho_{x,y} = 1 - \frac{6\sum_{i=1..|\mathcal{P}|} d_i^2}{|\mathcal{P}|(|\mathcal{P}|^2-1)}$, avec $d_i = r_{g_x}(\mathcal{P}, P_i) - r_{g_y}(\mathcal{P}, P_i)$ afin d'évaluer la précision globale du rangement ; **(ii)** la métrique de rappel $\texttt{R@k} = \frac{|\{rg_x(P_i)\ tq\ rg_x(P_i) \leq k\}|}{k} | i \in [1 \ldots, n]$, afin d'évaluer l'efficacité de notre approche pour identifier les k motifs les plus intéressants ; **(iii)** le temps CPU nécessaire à l'apprentissage.

Jeu de données : Nous avons sélectionné plusieurs jeux de données de taille réelle dans le référentiel FIMI [1]. Ces jeux de données ont des caractéristiques variées et représentent différents domaines d'application.

Les jeux de données sont présentés par ordre croissant de `#règ`.

		RankingSVM.0				AHPRank.0			
	#règ	ρ	R@10%	R@1%	t(s)	ρ	R@10%	R@1%	t(s)
Zoo	5×10^2	0,97	0,85	0,76	0,00	0,98	0,89	0,74	0,00
Vote	1×10^3	0,93	0,72	0,56	0,01	0,94	0,80	0,76	0,00
Anneal	5×10^3	0.99	0,89	0,81	0,10	0,87	0,85	0,4	0,00
Chess	1×10^4	0,78	0,83	0,78	1	0,72	0,50	0,30	0,01
Mushroom	2×10^4	0.99	0,97	0.93	2	0,97	0,61	0,34	0,02
Connect	5×10^4	0.92	0,81	0,62	18	0,75	0,91	0,82	0,04
T10	75×10^3	1	1	1	217	0,93	0,76	0,65	0,15
T40	1×10^5	0,99	0,99	0,98	72	0,99	0,95	0,60	0,27
Pumsb	15×10^5	0.93	0,99	0,92	77	0,91	0,97	0,93	0,43
Retail	2×10^5	0,95	0,97	0,95	250	0,93	0,82	0,80	0,80
moyenne		0,94	0,90	0,83	64	0,89	0,80	0,62	0,17

TAB. 1: Résultats de la validation croisée.

Résultats : Notre première expérience consiste à effectuer une validation croisée à 5 plis sur les `#règ` de chaque ensemble de données. Dans chaque pli, nous sélectionnons aléatoirement 20% des `#règ` pour former les données d'apprentissage et utilisons les 80% de règles restantes pour le test. Le pourcentage faible de la base d'apprentissage est motivé par notre contexte particulier de fouille interactive où les données utilisateurs sont de petite taille. Le tableau 1 contient les résultats de la validation croisée de `RankingSVM.0` et `AHPRank.0` en utilisant la corrélation de rang moyenne ρ, les valeurs de rappel (R@10% et R@1%) et le temps CPU en secondes sur les plis. Nous avons observé qu'étant donné une mesure unique, les résultats sont chaotiques. Par exemple, la mesure `Laplace` est fortement corrélée à χ^2 sur l'ensemble de données `T10` ($\rho = 99\%$) et faiblement corrélée lorsqu'elle est utilisée sur l'ensemble de

1. fimi.uantwerpen.be/data/

données Chess ($\rho = 0\%$). Cependant, la construction théorique de VBM[2] est d'une grande précision ($moyenne = 91\%$), ce qui nous amène à penser qu'une agrégation pondérée de ces mesures peut conduire à un bon compromis.
RankingSVM.0 **vs** AHPRank.0 : En terme de précision du rangement, nous observons que RankingSVM.0 surpasse AHPRank sur 8 jeux. La précision de RankingSVM.0 est très élevée et même parfaite en apprenant χ^2 sur T10 par exemple ($\rho = 100\%$). D'autre part, AHPRank.0 est très compétitif, et parfois meilleur que RankingSVM.0 surtout lorsque les données d'apprentissage sont de petite taille (voir Zoo et Vote). Les mêmes observations peuvent être faites en terme de rappel en haut du rangement à 10% et 1% (R@10% et R@1%). Cependant, la grande précision de RankingSVM.0 se fait au détriment du temps d'exécution, qui est de l'ordre de 4 minutes, alors que le temps nécessaire à AHPRank.0 pour apprendre ne dépasse pas une seconde.
RankingSVM.1 **vs** AHPRank.1 : Pour notre prochaine expérience, nous faisons une comparaison entre RankingSVM et AHPRank du point de vue de l'apprentissage actif. La première observation est que AHPRank.1 est plus performant que RankingSVM.1. En effet, les résultats de Mushroom montrent que AHPRank.1 est capable de classer 20 miles motifs avec une précision de 95% en utilisant seulement 200 requêtes (70% observé avec RankingSVM.1). En terme de temps de latence entre deux requêtes TQ , nous observons que RankingSVM.1 peut être gêné par une latence élevée même avec des requêtes de taille 2. Par exemple, sur le jeu de données T10, le temps de latence entre deux requêtes peut atteindre 8 minutes en moyenne, en traitant seulement 200 requêtes, alors que AHPRank.1 montre un comportement instantané (moins de 0.1 seconde).

6 Conclusion

En résumé, notre principale conclusion est que AHPRank offre un bon compromis entre la précision et les performances. AHPRank est capable d'atteindre une bonne précision globale de rangement en un temps linéaire par rapport à la taille des données d'entraînement, ce qui le rend adapté à une exploitation industrielle. Nous avons présenté un cadre générique et efficace pour l'apprentissage d'une fonction de rangement de motifs à l'aide d'une méthode multicritère basée sur AHP. L'algorithme AHPRank peut être appliqué dans les processus d'apprentissage passif et actif. L'approche proposée regroupe toutes les mesures en une seule fonction en utilisant une formulation linéaire pondérée, de sorte que le rangement induit soit aussi proche que possible du rangement de l'utilisateur. Nous avons appliqué ce cadre sur un cas d'étude de l'extraction de règles d'association.

Remerciements. Ce travail a reçu le soutien de l'Université de Montpellier et de l'I-Site MUSE dans le cadre du projet CAR-UM2020/2021.

Références

Agrawal, R., T. Imielinski, et A. N. Swami (1993). Mining association rules between sets of items in large databases. In *SIGMOD Conference*, pp. 207–216. ACM Press.

2. Correspond à la meilleure corrélation obtenue parmi l'ensemble des mesures dans $\mathcal{M}$.

Bie, T. D. (2011). An information theoretic framework for data mining. In *KDD*, pp. 564–572. ACM.

Boley, M., M. Mampaey, B. Kang, P. Tokmakov, et S. Wrobel (2013). One click mining : interactive local pattern discovery through implicit preference and performance learning. In *IDEA@KDD*, pp. 27–35. ACM.

Brunelli, M. (2015). *Introduction to the Analytic Hierarchy Process*. Springer.

Dzyuba, V. et M. van Leeuwen (2013). Interactive discovery of interesting subgroup sets. In *International Symposium on Intelligent Data Analysis*, pp. 150–161. Springer.

Dzyuba, V. et M. van Leeuwen (2017). Learning what matters - sampling interesting patterns. In *PAKDD (1)*, Volume 10234 of *Lecture Notes in Computer Science*, pp. 534–546.

Figueira, J., S. Greco, et M. Ehrgott (2005). *Multiple Criteria Decision Analysis : State of the Art Surveys*. Boston, Dordrecht, London : Springer Verlag.

Geng, L. et H. J. Hamilton (2006). Interestingness measures for data mining : A survey. *ACM Comput. Surv. 38*(3), 9.

Imielinski, T. et H. Mannila (1996). A database perspective on knowledge discovery. *Commun. ACM 39*(11), 58–64.

Kendall, M. G. et B. B. Smith (1939). The problem of m rankings. *Ann. Math. Statist. 10*(3), 275–287.

Raedt, L. D. (2002). A perspective on inductive databases. *SIGKDD Explor. 4*(2), 69–77.

Ringuest, J. L. (1986). A chi-square statistic for validating simulation-generated responses. *Comput. Oper. Res. 13*(4), 379–385.

Saaty, T. L. (1988). What is the analytic hierarchy process ? In *Mathematical models for decision support*, pp. 109–121. Springer.

Silberschatz, A. et A. Tuzhilin (1995). On subjective measures of interestingness in knowledge discovery. In *KDD*, pp. 275–281. AAAI Press.

Tan, P.-N., V. Kumar, et J. Srivastava (2004). Selecting the right objective measure for association analysis. *Information Systems 29*(4), 293–313.

Xin, D., X. Shen, Q. Mei, et J. Han (2006). Discovering interesting patterns through user's interactive feedback. In *KDD*, pp. 773–778. ACM.

Summary

The discovery of interesting patterns is a challenging task in data mining. On one hand, approaches have been proposed to automatically learn user-specific ranking functions over patterns. These approaches are often accurate, but very time consuming. On the other hand, many interest measures are used to evaluate the interest of patterns while being as close as possible to a user interest. In this paper, we express the learning of a pattern ranking function as a multicriteria optimization problem. The proposed approach aggregates all measures into a single weighted linear function, where coefficients are computed via the Analytic Hierarchy Process (AHP). Experiments conducted on many datasets show that our approach drastically reduces the execution time, while ensuring a high quality ranking compared to existing approaches.