CLASSIFICATION SUPERVISÉE PAR RÈGLES D'ASSOCIATIONS

Fouille de données avancées

Members:
Rudresh Mishra
Runlu Qu
Raymond Klutse
Ricardo M. Rodriguez O.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Date:
3 Mars , 2020

SOMMAIRE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

INTRODUCTION

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- La fouille de données est un processus d'analyse de données volumineuses afin d'en trouver les patterns utiles qui donnent l'information.

- Il est appliqué aux domaines tels que l'analyse du panier de marché, la gestion de la relation client (GRC) , la détection des fraudes etc.

- Pour ce faire, l'un des moyens est d'utiliser les **règles d'association**.

- C'est une méthode d'apprentissage non supervisé qui cherche les patterns fréquents et des corrélations parmi un ensemble d'éléments ou d'objets dans des bases de données.
- Algorithmes: Apriori, FP-Growth, Eclart...
- Exemple le plus courant: **Market Basket**

Market-Basket Transactions

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

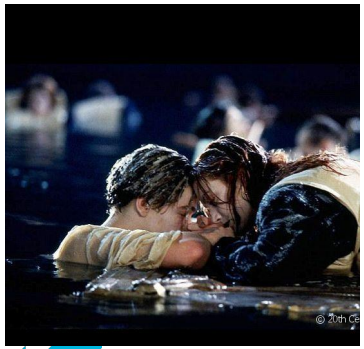
{Diaper} → {Beer}
{Milk, Bread} → {Eggs, Coke}
{Beer, Bread} → {Milk}

Fig 1. Analyse du panier de marché

- Cette relation implique la co-occurrence, et non la causalité

Source: http://www.zaxrosenberg.com/wp-content/uploads/2018/02/Association-Rule-Learning.pdf

- La classification associative (AC) est une approche prometteuse de fouille de données qui intègre les règles d'association et la classification supervisée pour construire des modèles de classification (classificateurs).
- Il considère uniquement le label de classe comme la conséquence d'une règle.
- Algorithmes: Classification based on Association (CBA), Classification based on multiple association rules (CMAR).
- Exemple:



```
##     lhs                                rhs            support
## [1] {Class=2nd,Age=Child}           => {Survived=Yes} 0.010904134
## [2] {Class=1st,Sex=Female}          => {Survived=Yes} 0.064061790
## [3] {Class=2nd,Sex=Female}          => {Survived=Yes} 0.042253521
## [4] {Class=Crew,Sex=Female}         => {Survived=Yes} 0.009086779
## [5] {Class=2nd,Sex=Male,Age=Adult} => {Survived=No}  0.069968196
## [6] {Class=2nd,Sex=Male}           => {Survived=No}  0.069968196
## [7] {Class=3rd,Sex=Male,Age=Adult} => {Survived=No}  0.175829169
## [8] {Class=3rd,Sex=Male}           => {Survived=No}  0.191731031
```

Fig 2. Jeu de données Titanic

Source: http://www.zaxrosenberg.com/wp-content/uploads/2018/02/Association-Rule-Learning.pdf

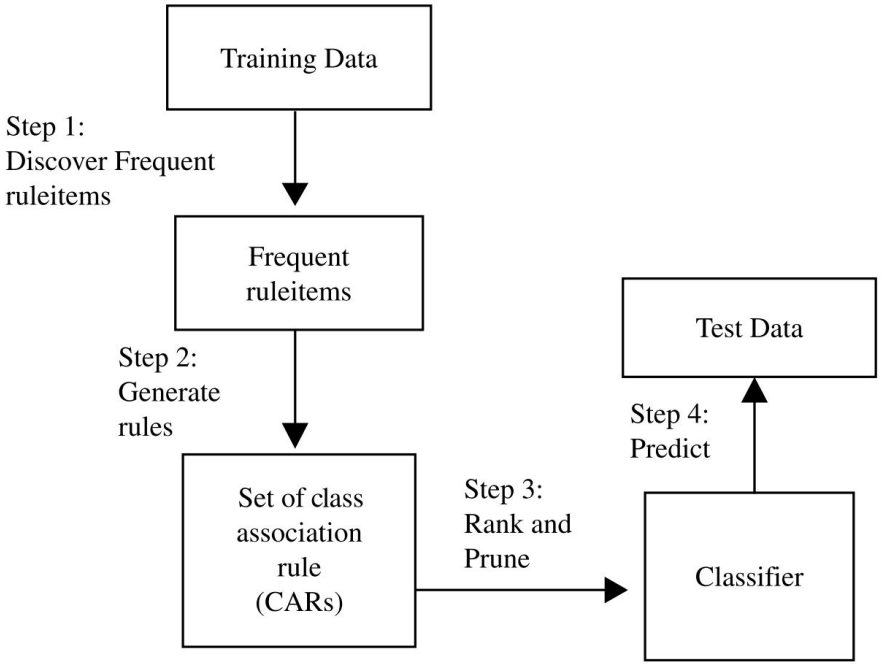


Fig 3. Schéma de Classification associative [3]

- Génération de règle

- Construction du classificateur

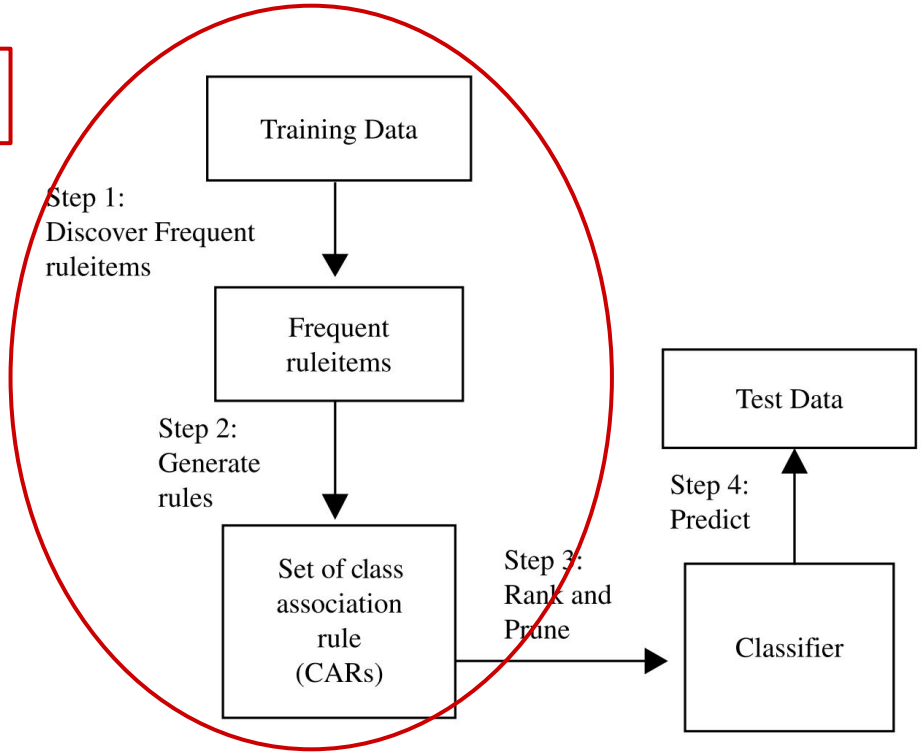


Fig 3. Schéma de Classification associative [3]

- Génération de règle
- Construction du classificateur

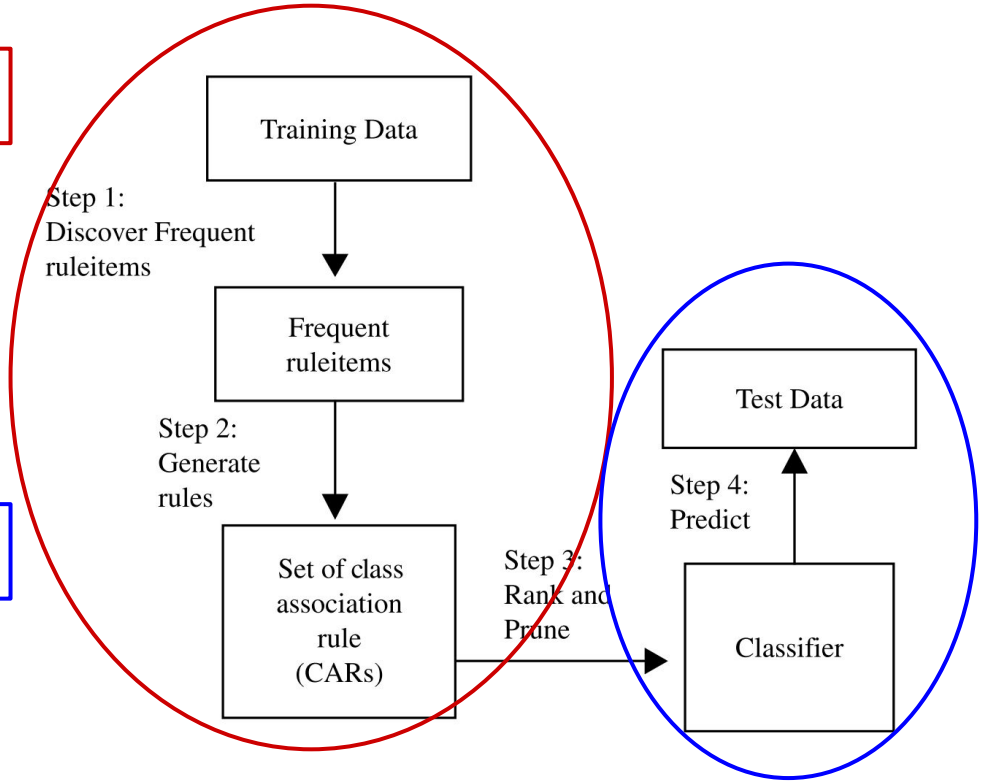


Fig 3. Schéma de Classification associative [3]

Types de méthodes de classification associatives :

- **CBA (Classification based on Associations:Liu, Hsu & Ma, KDD'98)**
  - Les règles d'association sous forme *un ensemble de paires de valeurs d'attributs -> class label*
  - Un classificateur est construit en organisant les règles en fonction d'une priorité décroissante basée sur la Confidence et le Support.

- **CMAR (Classification based on multiple association rules: Li, Han, Pei, ICDM'01)**
  - Un classificateur est construit en utilisant plusieurs les règles au lieu d'un seul.

- **CPAR (Classification based on Predictive Association Rules: Yin & Han, SDM'13)**
  - Génération de règles prédictives (analyse de type FOIL) qui permet de conserver les règles couvertes avec des poids réduits.
  - Haute efficacité, précision similaire à la CMAR

ALGORITHMES

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Pour introduire les algorithmes, nous rêvons sur les notions de base:

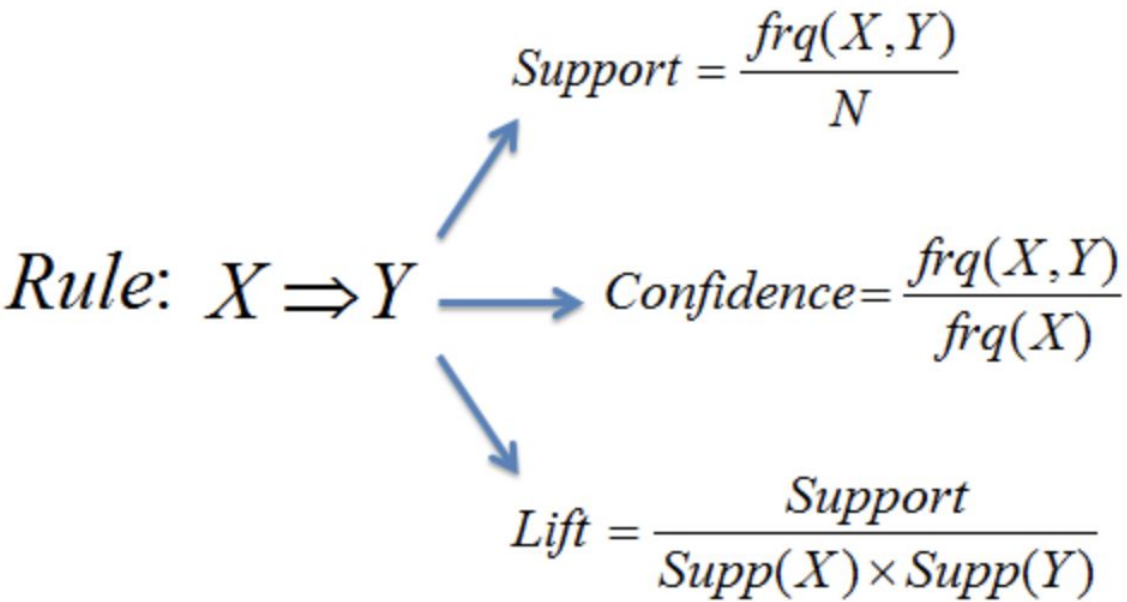


Fig 4. Terminologies de base [4]

2. ALGORITHMES

2.2 Classification Based on Associations (CBA) [4]

- Un des algorithmes le plus utilisé de la famille “Classification Associative” (CA)

- Il existe deux versions de cet algorithme : M1 et M2, qui sont différents en vitesse mais produisent normalement les mêmes résultats.

- Phases:
  1. La discrétisation des attributs continus, s’il existe des cas.
  2. Générer toutes les Class Association Rule (**CARs**) (Rule Generation)
  3. La construction d'un classificateur basé sur les **CARs** générés (Classifier Builder)

1) Discretization des attributs

- Attribut **catégorique** -> un ensemble d'**entiers positifs consécutifs**
- Attribut **continu** -> discrétisée en **intervalles** -> des **entiers positifs consécutifs**.

- A la fin un cas de données est vu comme un ensemble de paires (attributs, valeurs entières) avec une étiquette de classe. Ex:

1. <{(A, 1), (B, 1)}, (class: 1)>.
2. <{(A, 1), (B, 1)}, (class: 2)>.

2) Generation des régles

- On utilise le algorithm Apriori
- On sélectionne ceux qui ont la class au droite

Algorithme M1 - Classifier Builder

Objectifs:

- Évaluer tous les sous-ensembles possibles sur les données de formation
- Sélectionner le sous-ensemble qui donnent le **moins d'erreurs possible.**

```
1  R = sort(R);
2  foreach rule r ∈ R do
3      temp = ∅;
4      foreach data case d ∈ D do
5          if d satisfies the conditions of r then
6              store d.id in temp and mark r if it correctly classifies d
7          end
8      end
9      if r is marked then
10         insert r at the end of C;
11         delete all the cases with the ids in temp from D;
12         select a default class for the current C;
13         compute the total number of errors of C;
14     end
15 end
16 Find the first rule p in C with the lowest total number of errors and drop
     all the rules after p in C;
17 Add the default class associated with p to the end of C and return C;
```

[7]

Algorithme M1 - Classifier Builder

- Dans un premier temps, toutes les règles sont triées en fonction de la “confidence” et “support”

```
1  R = sort(R);
2  foreach rule r ∈ R do
3      temp = ∅;
4      foreach data case d ∈ D do
5          if d satisfies the conditions of r then
6              store d.id in temp and mark r if it correctly classifies d
7          end
8      end
9      if r is marked then
10         insert r at the end of C;
11         delete all the cases with the ids in temp from D;
12         select a default class for the current C;
13         compute the total number of errors of C;
14     end
15 end
16 Find the first rule p in C with the lowest total number of errors and drop
     all the rules after p in C;
17 Add the default class associated with p to the end of C and return C;
```

[7]

Algorithme M1 - Classifier Builder

- Si une instance satisfait à la règle et que la classe de l'instance correspond au résultat, la règle est marquée et insérée dans le fichier temp.

```
1  R = sort(R);
2  foreach rule r ∈ R do
3      temp = ∅;
4      foreach data case d ∈ D do
5          if d satisfies the conditions of r then
6              store d.id in temp and mark r if it correctly classifies d
7          end
8      end
9      if r is marked then
10         insert r at the end of C;
11         delete all the cases with the ids in temp from D;
12         select a default class for the current C;
13         compute the total number of errors of C;
14     end
15 end
16 Find the first rule p in C with the lowest total number of errors and drop
     all the rules after p in C;
17 Add the default class associated with p to the end of C and return C;
```

[7]

Algorithme M1 - Classifier Builder

- Si la règle est marquée, elle est insérée à la fin du classificateur et toutes les instances en temp sont retirées de l'ensemble de données.
- La classe par défaut est sélectionnée (la classe majoritaire des instances restantes).
- Ensuite, on évalue les erreurs sont commises par un classificateur C.
- Le nombre d'erreurs est associé à la règle en cours.

```
1  R = sort(R);
2  foreach rule r ∈ R do
3      temp = ∅;
4      foreach data case d ∈ D do
5          if d satisfies the conditions of r then
6              store d.id in temp and mark r if it correctly classifies d
7          end
8      end
9      if r is marked then
10         insert r at the end of C;
11         delete all the cases with the ids in temp from D;
12         select a default class for the current C;
13         compute the total number of errors of C;
14     end
15 end
16 Find the first rule p in C with the lowest total number of errors and drop
     all the rules after p in C;
17 Add the default class associated with p to the end of C and return C;
```

[7]

Algorithme M1 - Classifier Builder

- La règle r avec le plus petit nombre d'erreurs associé est trouvée, les règles en dessous de la règle r sont rejetées.
- La classe par défaut associée à la règle r est alors ajoutée en dessous de r comme dernière règle du classificateur final C.

```
1  R = sort(R);
2  foreach rule r ∈ R do
3      temp = ∅;
4      foreach data case d ∈ D do
5          if d satisfies the conditions of r then
6              store d.id in temp and mark r if it correctly classifies d
7          end
8      end
9      if r is marked then
10         insert r at the end of C;
11         delete all the cases with the ids in temp from D;
12         select a default class for the current C;
13         compute the total number of errors of C;
14     end
15 end
16 Find the first rule p in C with the lowest total number of errors and drop
     all the rules after p in C;
17 Add the default class associated with p to the end of C and return C;
```

[7]

- Cette méthode s’étend l'algorithme de FP-Growth en construisant un arbre FP associé à une distribution de classes afin d'exploiter efficacement la base de données et de rendre le processus évolutif.
- Il comprend 2 phases:
  - **Génération de règles**
  - **Classification**
- Dans la phase de génération des règles, la CMAR parcourt le jeu de données pour en trouver l'ensemble complet des règles qui dépassent certains seuils de support et de confidence.

- Les éléments dans le jeu de données avec support supérieur ou égal au seuil de support défini sont placés dans un ensemble connu sous le nom d'ensemble d'éléments fréquents, F.

- Ensuite, la CMAR trie les valeurs dans la liste F dans l'ordre décroissant et parcourt à nouveau le jeu de données pour construire un arbre FP, comme le montre la figure.

Row-id	A	B	C	D	Class label
1	a_1	b_1	c_1	d_1	A
2	a_1	b_2	c_1	d_2	B
3	a_2	b_3	c_2	d_3	A
4	a_1	b_2	c_3	d_3	C
5	a_1	b_2	c_1	d_3	C

Fig 5. Jeu de données [1]

Seuil de Support : 2

Seuil de Confidence : 50 %

Attribut	Support
a1	4
b2	3
c1	3
d3	3
a2	1
b1	1
b3	1
c2	1
c3	1
d1	1
d2	1

$$F = \{a_1, b_2, c_1, d_3\}$$

Row-id	Sort_per_F_list	Class Label
1	(a1,c1)	A
2	(a1,b2,c1)	B
3	(d3)	A
4	(a1,b2)	C
5	(a1,b2,c1)	C

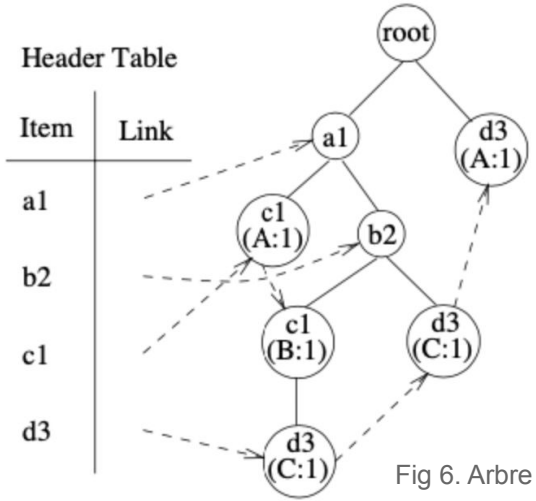


Fig 6. Arbre de FP [1]

- Une fois qu'une règle est générée, elle est stockée dans un arbre de CR.

Rule-id	Rule	Support	Confidence
1	$abc \rightarrow A$	80	80%
2	$abcd \rightarrow A$	63	90%
3	$abe \rightarrow B$	36	60%
4	$bcd \rightarrow C$	210	70%

Fig 7. Les règles du jeu de données [1]

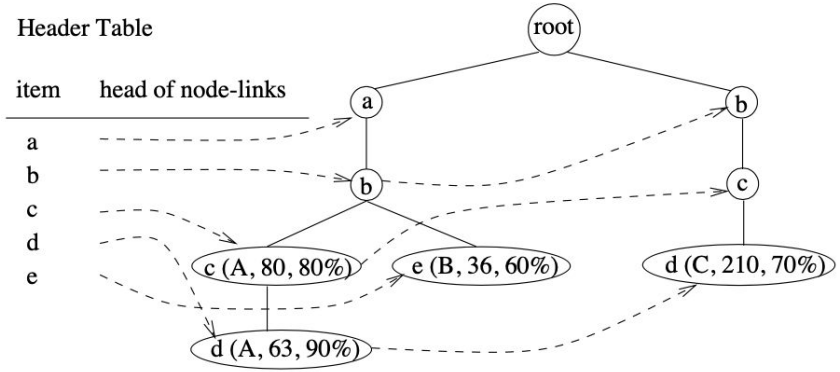


Fig 8. CR-tree pour les règles [1]

- Le nombre de règles générées par la fouille des règles d'association de classes peut être énorme ainsi ils sont élaguées avant de passer à la prochaine étape.

- Dans la phase de classification, la CMAR classifie un nouvel objet en rassemblant un sous-ensemble de règles correspondant au nouvel objet dans l'ensemble des règles de classification
- Trivialement, si toutes les règles correspondant au nouvel objet ont la même étiquette de classe, la CMAR attribue simplement cette étiquette au nouvel objet.
- Si les règles ne sont pas cohérentes dans les étiquettes de classe, la CMAR divise les règles en groupes en fonction des étiquettes de classe, où toutes les règles d'un groupe partagent la même étiquette de classe et où chaque groupe a une étiquette distincte.

- Chaque groupe a plusieurs règles avec la même classe et le CMAR compare les effets des groupes en mesurant “l’effet combiné" de chaque groupe.
- Si les règles d'un groupe sont fortement corrélées positivement et bénéficient d'un bon Support, le groupe devrait avoir un effet fort.
- Les méthodes disponibles pour mesurer l'effet combiné d'un groupe de règles sont les suivantes :
  - **En utilisant la mesure du chi carré (X^2)**
  - **Utilisation du composé de corrélation des règles**
- Chacune de ces méthodes a ses défauts ; c'est pourquoi on utilise la mesure **X^2 pondérée** qui intègre à la fois les deux mesures.
- Le groupe ayant la valeur la plus élevée est considéré comme le plus fort et est affecté au cas test [2].

Data set	# attr	# cls	# rec	C4.5	CBA	*CMAR*
Anneal	38	6	898	94.8	97.9	97.3
Austral	14	2	690	84.7	84.9	86.1
Auto	25	7	205	80.1	78.3	78.1
Breast	10	2	699	95	96.3	96.4
Cleve	13	2	303	78.2	82.8	82.2
Crx	15	2	690	84.9	84.7	84.9
Diabetes	8	2	768	74.2	74.5	75.8
German	20	2	1000	72.3	73.4	74.9
Glass	9	7	214	68.7	73.9	70.1
Heart	13	2	270	80.8	81.9	82.2
Hepatic	19	2	155	80.6	81.8	80.5
Horse	22	2	368	82.6	82.1	82.6
Hypo	25	2	3163	99.2	98.9	98.4
Iono	34	2	351	90	92.3	91.5
Iris	4	3	150	95.3	94.7	94

Fig 9. Comparaison de l'accurraccy de 4.5, CBA et CMAR[1]

DÉMONSTRATION DE CBA

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

4.DÉMONSTRATION DE CBA

Exemple dans Google Cloud

DISCUSSION

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

2. DISCUSSION
Comparison with association rules

Règles d'association	Classification associative
➢ Aucun attribut de classe impliqué (apprentissage non supervisé)	➢ Une classe doit être donnée (apprentissage supervisé)
➢ L'objectif est de découvrir l'association entre les éléments d'une base de données transactionnelle	➢ L'objectif est de construire un classificateur qui puisse prédire les classes des objets de données de test.
➢ Il peut y avoir plus d'un attribut dans la conséquence d'une règle	➢ Il n'y a qu'un attribut (attribut de classe) dans la suite d'une règle
➢ Overfitting n'est généralement pas un problème	➢ Overfitting est une question importante

Difference entre règles d'association et association classification [3]

2. DISCUSSION
Comparison with association rules

Règles d'association	Classification associative
➢ Aucun attribut de classe impliqué (apprentissage non supervisé)	➢ Une classe doit être donnée (apprentissage supervisé)
➢ L'objectif est de découvrir l'association entre les éléments d'une base de données transactionnelle	➢ L'objectif est de construire un classificateur qui puisse prédire les classes des objets de données de test.
➢ Il peut y avoir plus d'un attribut dans la conséquence d'une règle	➢ Il n'y a qu'un attribut (attribut de classe) dans la suite d'une règle
➢ Overfitting n'est généralement pas un problème	➢ Overfitting est une question importante

Difference entre règles d'association et association classification [3]

Pourquoi nous combinons les règles d'association et la classification ensemble?

Pourquoi nous combinons les règles d'association et la classification ensemble?
Avantage: ils sont capables d'utiliser les règles de classification les plus précises car leurs générateur de règles ont pour objectif de trouver toutes les règles.

Pourquoi nous combinons les règles d'association et la classification ensemble?
Avantage: ils sont capables d'utiliser les règles de classification les plus précises car leurs générateur de règles ont pour objectif de trouver toutes les règles.

Est-ce le meilleur système de classification?

Pourquoi nous combinons les règles d'association et la classification ensemble?
Avantage: ils sont capables d'utiliser les règles de classification les plus précises car leurs générateur de règles ont pour objectif de trouver toutes les règles.

Est-ce le meilleur système de classification?

- L'exploration de règles d'association traditionnelle n'utilise qu'un seul *minsup* dans la génération de règles, ce qui est inadéquat pour une distribution de classe déséquilibrée.
- Les données de classification contiennent souvent un grand nombre de règles, ce qui peut provoquer une explosion combinatoire. Pour de nombreux jeux de données, le générateur de règles est incapable de générer des règles avec de nombreuses conditions, alors que ces règles peuvent être importantes pour une classification précise.[5]

Comment résoudre?

Le premier problème est traité en utilisant plusieurs supports de classe minimum,

tandis que le deuxième problème est traité en le combinant avec d'autres méthodes de classification, la méthode de l'arbre de décision étant particulièrement efficace.[5]

CBA(2) + C4.5 + NB Vs									
CBA(2)	CBA	C4.5 tree	C4.5 rules	RIPPER	NB	LB	C4.5 + NB	CBA(2) + NB	CBA(2) + C4.5

REFERENCES

[1] Li.,W et al, “CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules”, Canada, 2001.

[2] Wedyan,S., “Review and Comparison of Associative Classification Data Mining Approaches”, World Academy of Science, Engineering and Technology International Journal of Industrial and Manufacturing Engineering, vol.8, no.1, 2014.

[3] Thabtah,F. ,“A review of associative classification mining”, The Knowledge Engineering Review Cambridge University Press , Vol. 22:1, 37–65, 2007.

[4] Liu, B., Hsu, W., & Ma, Y. Integrating classification and association rule mining In KDD, vol. 98, pp. 80-86, 1998.

[5]Liu, B., Ma, Y., Wong, C.K.: Classification using association rules: weaknesses and enhancements. Data mining for scientific applications 591,2001.

[6] Sakar , P.K , “Association rule learning: A brief overview [Part-1]”, Accessed : 2nd March 2020, Available on: https://medium.com/data-science-vibes/association-rule-part-1-f37e3cc545a0

[7]Filip Jiˇr´ı and Tom´aˇs Kliegr, “Classification based on Associations (CBA) - a performance analysis”

MERCI POUR VOTRE ATTENTION

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

ANNEXE

Classification Based on Associations (CBA) [4]

Algorithme M1 - Rule Generation

- Premiere occurrence de l'algorithme
- Il compte les occurrences des items et des classes pour déterminer les l-rule items fréquentes (ligne 1).
- Les CARs sont générés et une opération de pruning est ensuite effectuée (cela peut être facultatif)

```
1   F_1 = {large 1-ruleitems};
2   CAR_1 = genRules(F_1);
3   prCAR_1 = pruneRules(CAR_1);
4   for (k = 2; F_{k-1} ≠ ∅; k++) do
5       C_k = candidateGen(F_{k-1});
6       for each data case d ∈ D do
7           C_d = ruleSubset(C_k, d);
8           for each candidate c ∈ C_d do
9               c.condsupCount++;
10              if d.class = c.class then c.rulesupCount++
11          end
12      end
13      F_k = {c ∈ C_k | c.rulesupCount ≥ minsup};
14      CAR_k = genRules(F_k);
15      prCAR_k = pruneRules(CAR_k);
16  end
17  CARs = ⋃_k CAR_k;
18  prCARs = ⋃_k prCAR_k;
```

Figure 1: The CBA-RG algorithm [7]

Each *k* element of this set is of the following form: <(*condset*, *condsupCount*), (*y*, *rulesupCount*)>.

ANNEXE

Classification Based on Associations (CBA) [4]

Algorithme M1 - Rule Generation

- La phase de **création des règles**:

- **Objectif:** générer l'ensemble complet des CARs qui satisfont aux contraintes de support minimum (**minsup**) et de confiance minimum (**minconf**) spécifiées par l'utilisateur.

- The set of class association rules (**CARs**) thus consists of all the Possible Rules (PRs) that are both frequent and accurate.

1 $F_1 = \{\text{large 1-ruleitems}\}$;
2 $CAR_1 = \text{genRules}(F_1)$;
3 $prCAR_1 = \text{pruneRules}(CAR_1)$;
4 **for** $(k = 2; F_{k-1} \neq \emptyset; k{+}{+})$ **do**
5 $C_k = \text{candidateGen}(F_{k-1})$;
6 **for** each data case $d \in D$ **do**
7 $C_d = \text{ruleSubset}(C_k, d)$;
8 **for** each candidate $c \in C_d$ **do**
9 c.condsupCount++;
10 **if** d.class = c.class **then** c.rulesupCount++
11 **end**
12 **end**
13 $F_k = \{c \in C_k \mid c.\text{rulesupCount} \geq minsup\}$;
14 $CAR_k = \text{genRules}(F_k)$;
15 $prCAR_k = \text{pruneRules}(CAR_k)$;
16 **end**
17 $CARs = \bigcup_k CAR_k$;
18 $prCARs = \bigcup_k prCAR_k$;

Figure 1: The CBA-RG algorithm [7]

ANNEXE

Classification Based on Associations (CBA) [4]

Algorithme M1 - Rule Generation

- La fonction “candidateGen” est similaire a l’algorithme Apriori

- For each subsequent pass, say pass k, the algorithm performs 4 major operations. First, the frequent ruleitems F found in the (k-1)th pass are used to generate thek-1 candidate ruleitems C using the condidateGen functionk (line 5)

```
1   F_1 = {large 1-ruleitems};
2   CAR_1 = genRules(F_1);
3   prCAR_1 = pruneRules(CAR_1);
4   for (k = 2; F_{k-1} ≠ ∅; k++) do
5       C_k = candidateGen(F_{k-1});
6       for each data case d ∈ D do
7           C_d = ruleSubset(C_k, d);
8           for each candidate c ∈ C_d do
9               c.condsupCount++;
10              if d.class = c.class then c.rulesupCount++
11          end
12      end
13      F_k = {c ∈ C_k | c.rulesupCount ≥ minsup};
14      CAR_k = genRules(F_k);
15      prCAR_k = pruneRules(CAR_k);
16  end
17  CARs = ⋃_k CAR_k;
18  prCARs = ⋃_k prCAR_k;
```

Let *Ck* be the set of candidate *k-ruleitems*

Figure 1: The CBA-RG algorithm [7]