TP 4 - Règles d'association avec WEKA

Rappel Modalité : Vous effectuerez un compte rendu écrit des TPs que vous effectuerez en Data Mining. Ce compte rendu pourra être fait en binôme. Les éléments obligatoires apparaîtront au fur et à mesure des TP, ces éléments ne sont pas limitatifs et nous vous encourageons vivement à aller plus loin (interprétation des résultats, …)

Jeux de données : Les jeux de données avec lesquels vous allez travailler sont ceux notamment fournis par l'UCI http://www1.ics.uci.edu/~mlearn/MLSummary.html et sont disponibles sur http://www.lifl.fr/~jourdan/FDD/

1. Un premier exemple de règle d'association

1. Lancez Weka, puis l'Explorer. Choisissez le fichier weather.nominal.arff : c'est l'exemple standard du golf (ou du tennis. . .), où tous les attributs ont été discrétisés. Les algorithmes de recherche de règles d'association se trouvent sous l'onglet *Associate.*
2. Choisissez l'algorithme Apriori.
3. Vérifiez que tout fonctionne en lançant l'algorithme sans modifier les paramètres du programme.
4. Quelles sont les informations retournées par l'algorithme ?

2. Modification des paramètres

En cliquant du bouton droit dans la fenêtre en face du bouton *Choose*, on a accès aux paramètres de l'algorithme.

Le bouton More détaille chacune de ces options.

delta : fait décroître le support minimal de ce facteur, jusqu'à ce que soit le nombre de règles demandées a été trouvé, soit on a atteint la valeur minimale du support lowerBoundMinSupport

lowerBoundMinSupport : valeur minimale du support (minsup en cours). Le support part d'une valeur initiale, et décroît conformément à delta.

metricType : la mesure qui permet de classer les règles. Supposons la règle L→ R où L désigne la partie gauche de la règle et R la partie droite. Il y en a quatre :

- Confidence : la confiance.
- Lift : l'amélioration (cf cours).
  ***lift* = P(*L*,*R*) / P(*L*).P(*R*)**.
- leverage : proportion d'exemples concernés par les parties gauche et droite de la règle, en plus de ce qui seraient couverts, si les deux parties de la règles étaient indépendantes
  ***leverage* = P(*L*,*R*) - P(*L*).P(*R*)**.
- Conviction : similaire à l'amélioration, mais on s'intéresse aux exemples où la partie droite de la règle n'est pas respectée.
  ***conviction* = Pr(*L*).Pr(not *R*) / Pr(*L*,not *R*)**.

minMetric : la valeur minimale de la mesure en dessous de laquelle on ne recherchera plus de règle.

numRules : Le nombre de règles que l'algorithme doit produire.

removeAllMissingCols : enlève les colonnes dont toutes le valeurs sont manquantes.

significanceLevel : test statistique

upperBoundMinSupport : valeur initiale du support.

1. Sur le fichier weather.nominal.arff, comparer les règles produites selon la mesure choisie.
2. Pour une règle que vous choisirez, vérifiez les calculs de *confiance*, *amélioration*, *leverage* et *conviction*.
3. Pouvez-vous décrire d'une phrase ce que signifient les notions de leverage et de conviction ?

3. Un deuxième exemple de règles d'association

Le fichier bank-data.cvs contient des données extraites d'un recensement de la population américaine. Le but de ces données est initialement de prédire si quelqu'un gagne plus de 50.000 dollars par an. On va d'abord transformer un peu les données :

a. Transformation des données

Récupérer le fichier bank-data.csv Revenez à la fenêtre Preprocess.
Tout d'abord ouvrez le fichier bank-data.csv (voir http://www.lifl.fr/~jourdan/FDD/)

1. Weka met à votre disposition des filtres permettant soit de choisir de garder (ou d'écarter) certains exemples, soit de modifier, supprimer, ajouter des attributs. La zone de sélection *Filters* vous permet de manipuler les filtres. Le fonctionnement général est toujours le même.
2. Pouvez-vous lancer l'algorithme Apriori ? Pourquoi ?

b. Sélection des attributs

Les données comportent souvent des attributs inutiles : numéro de dossier, nom,date de saisie . . . . Il est possible d'en supprimer 'à la main', à condition de connaître le domaine. On peut aussi lancer un algorithme de data mining, et regarder les attributs qui ont été utilisés : soient ceux-ci sont pertinents, et il est important de les garder, soient ils sont tellement liés à la classe qu'à eux seuls ils emportent la décision (pensez à un attribut qui serait la copie de la classe). Weka a automatisé cette recherche des attributs pertinents dans le filtre AttributeSelectionFilter, qui permet de définir les attributs les plus pertinents selon plusieurs méthodes de recherche (search), en utilisant plusieurs mesures possibles de la pertinence d'un attribut (eval).

1. Ici l'attribut id est une quantité qu'on peut ignorer pour la fouille : supprimez le !

c. Discrétisation

Certains algorithmes ont besoin d'attributs discrets pour fonctionner, d'autres n'acceptent que des attributs continus (réseaux de neurones, plus proches voisins). D'autres encore acceptent indifféremment des attributs des deux types. Weka dispose de filtres pour discrétiser des valeurs continues. Le *filtre DiscretizeFilter* permet de rendre discret un attribut continu et ceci de plusieurs façons :

- En partageant l'intervalle des valeurs possibles de l'attribut en intervalles de taille égale.
- En le partageant en intervalles contenant le même nombre d'éléments.
- En fixant manuellement le nombre d'intervalles (bins).
- En laissant le programme trouver le nombre idéal de sous intervalles.

Ici il y a plusieurs attributs numériques : "children", "income", "age".

1. Discrétiser age et income en utilisant le filtre Weka et en forçant le nombre d'intervalles à 3. Sauver le fichier transformé par exemple dans bank1.arff.
2. L'attibut children est numérique mais ne prend que 4 valeurs : 0,1,2,3 ; pour le discrétiser, on peut soit utiliser le filtre, soit le faire à la main dans le fichier arff.

Remarque : si vous éditez directement le fichier, vous pouvez en profiter pour rendre les données plus lisibles, par exemple en traduisant le nom des attributs, en donnant des noms aux intervalles obtenus par la discrétisation...

d. APriori

Sauvez dans « bank.arff » le résultat de vos transformations : c'est le fichier qui va servir pour la génération des règles d'association.

1. Appliquez l'algorithme Apriori et tentez d'interpréter les règles produites. Jouez sur les paramètres. Comment se comporte le temps d'exécution en fonction des paramètres ? Quels sont les paramètres les plus "critiques" ?
2. Utilisez l'algorithme Tertius. Que constatez-vous sur la forme des règles ?

4. Promotions de Pâques et épicerie de nuit

L'épicerie de nuit de la rue *"Campus Joyeux"* a décidé à l'approche des fêtes de pâques de lancer une vaste opération de promotion. Son patron, fervent adepte des nouvelles technologies et de la fouille de données (ça arrive), vous demande d'utiliser les règles d'associations pour trouver des règles intéressantes pour ses futures promotions. Il va donc réutiliser le bilan d'achats de l'année dernière à la même date :

Achats	Produit 1	Produit 2	Produit 3	Produit 4	Produit 5
Mme Michou	X			X	X
Tonton Gérard	X	X			X
Mme Guénolet					X
Mr Robert			X	X	X
Mr Sar	X	X	X	X	X
Mr causy	X				X
Mme Mimi	X			X	X
Mme Fillon		X	X		

Table 1 – Table d'achats de l'année 2013 (Pâques)

1. Générer un fichier ARFF contenant les données du bilan d'achat
2. Extraire les règles d'associations avec un support de 0.5 puis de 0.1
3. Que pouvez-vous conseiller comme promotion au patron ?

5. API Weka (Bonus)

Il est possible d'utiliser directement les algorithmes sur des jeux de données en les appelant directement à partir de votre propre code Java.

1. Etudiez l'API de Weka, notamment pour les règles d'Association, puis dans un programme Java, automatisez directement ce que vous avez fait via l'interface graphique en appelant directement les algorithmes nécessaires.

2. Utilisez le code précédent pour étudier le temps d'exécution de l'algorithme Apriori en fonction du seuil de support et du seuil de confiance (vous pouvez générer des graphes à l'aide de gnuplot).