Data Mining, Machine Learning, Apprentissage :une introduction avec le logiciel WEKA (langage Java)

Tout le matériel nécessaire se trouve soit sur http://www.math-info.univ-paris5.fr/~lomn/Cours/DM/Material/Tutorial/ (site web 1) soit sur http://www.cs.waikato.ac.nz/ml/weka/ (site web 2).

Nous travaillerons sous Linux même si le logiciel fonctionne identiquement sous Windows et évidemment *MacOS*. Nous utiliserons le logiciel *WEKA* développé par l'université de Waikato, New Zealand. Ce logiciel est un petit laboratoire portable pour tester, expérimenter et même coder en Java.

Prendre le logiciel Weka en main

D'abord, il faut l'installer.

Cette installation suppose qu'un JRE (Java RunTime Environment est installé sur votre machine). Il vient généralement par défaut avec les systèmes d'exploitations. Toutefois, si votre machine n'en possède pas se référer à http://rsb.info.nih.gov/ij/docs/install/linux.html et à la version plus complète qui contient un JRE (http://rsb.info.nih.gov/ij/download/linux/ij148-linux64.zip). Pour les futurs développeurs, un environnement tel que Java Development Kit (JDK) est conseillé (http://www.oracle.com/technetwork/java/javase/downloads/index.html). Cet environnement fournira notamment la librairie permettant (facilement) de compiler des plugins imageJ en Java pour les modifier. Nous verrons cela par la suite.

En résumé pour travailler avec Weka, il faut a minima un JRE. Avec un JDK on peut faire du développement (coder, compiler) et donc faire évoluer le logiciel assez facilement.

Astuce : pour vérifier la présence de jre, jdk et leur localisation sur vore système utiliser les commandes Unix : soit ***$locate *jre**** soit ***$find /usr -name *jre**** par exemple et à adapter.

Lancer le logiciel WEKA :

$java -Xmx1000M -jar weka.jar

Ce logiciel est développé en Java, et fournit le code source et la documentation complète des classes. On peut ainsi réutiliser les différents outils dans une application personnelle en important les classes nécessaires. Sachant que le langage JAVA propose des paquetages spécifiques à la gestion de bases de données, c'est un environnement idéal pour combiner les connaissances acquises en base de données, fouille de données et en programmation objet dans le cadre d'un stage ou d'un projet par exemple.

Prise en main et Étude de fleurs.

Nous allons analyser des données concernant de la botanique. Une donnée sera un échantillon pour nous.

Nous allons utiliser l'environnement *Explorer* de Weka.

L'interface est composée de plusieurs panneaux. Dans la suite, je fais confiance à votre perspicacité pour répondre à certaines questions grâce à l'interface proposée.

Le panneau *Preprocess*

Cette partie permet de retoucher les données pour les adapter à un format adéquate à la méthode de *data mining* que l'on sélectionnera plus tard.

Chargez le fichier de données ***iris.arff*** (à vous de le trouver, c'est un classique)

Combien y a-t-il d'attributs ? Quel est leur nom ?

Quelles sont les statistiques de répartition pour la largeur du pétale de chacune des fleurs analysés ?

De quel type sont chacune des données : binaires, nominal, numérique ?

Combien y a-t-il de données ?

On peut retoucher les données à ce niveau là. Par exemple, décider de ne pas utiliser un des attributs. Si l'on décoche ***petallength*** par exemple.

Versions de Xeka antérieures : Pour rendre effectif le changement, il faut cliquer sur ***Apply Filter***. Observez les modifications dans la zone ***Working Relation.***

Version de Weka récente : on utilise l'option Remove tout simplement.

Revenez à l'usage de tous les attributs.

Quelles sont les trois classes d'iris répertoriées par les botanistes en fonction des 4 attributs métriques stockés ?

Nous reviendrons à ce panneau pour appliquer d'autres filtres avec d'autres jeux de données.

Le panneau Visualisation

Ce panneau permet de visualiser les données par projection 2D.

En modifiant les attributs projetés sur l'axe des X et l'axe des Y, déterminez un couple d'attributs visuellement optimal pour séparer les données dans les trois classes précisées par les botanistes.

A droite de la zone de projection, se trouve 5 bandes. Elles correspondent aux 5 attributs stockés pour chaque fleur dont le nom de l'espèce ou classe. En cliquant avec les boutons de la souris sur ces bandes, on modifie les sélections en X et Y pour la visualisation en 2D.

Le panneau Select Attributes

Ce panneau permet de faire de la sélection d'attributs pertinents dans un objectif de classification. C'est ici que vous pouvez trouver un moyen automatique de déterminer le meilleur couple d'attributs pour séparer les classes.

En laissant les modalités définies par défaut dans ce panneau, lancez la sélection d'attributs en cliquant sur *Start*. Quelle est la réponse fournie par l'algorithme ?

Est-ce en accord avec votre réponse ?

Visualiser la solution proposée pour vous en persuader.

Le panneau Classify

Pour prendre contact, nous allons réaliser une classification basique. L'algorithme *ZeroR* détermine la classe la plus présente dans l'ensemble d'apprentissage et génère une règle unique du type
: classe 1

Qui se lit, si n'importe quoi pour les attributs 1à 4 alors conclure que la fleur est de la classe 1.

Sélectionnez cet algorithme en cliquant sur la liste déroulante *Classifier*. Aucun paramètre n'est à préciser.

Lancer l'algorithme en cliquant *Start*.

Dans la zone Classifier Output, les résultats sont affichés. Après avoir rappelé quelques informations sur le nom de la relation (au sens des bases de données relationnelles) et sa structure, puis sur le type d'algorithme d'analyse utilisé, la partie *Summary* détaille les résultats en classification dont nous présentons l'essentiel dans ce qui suit.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          50               33.3333 %
Incorrectly Classified Instances       100               66.6667 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===
```

TP Rate	FP Rate	Precision	Recall	Class
1	1	0.333	1	Iris-setosa
0	0	0	0	Iris-versicolor
0	0	0	0	Iris-virginica

=== Confusion Matrix ===

a	b	c	<-- classified as
50	0	0	a = Iris-setosa
50	0	0	b = Iris-versicolor
50	0	0	c = Iris-virginica

D'abord, il y a la façon dont on valide le résultat de l'algorithme : le plus souvent on utilise la technique de cross-validation. C'est-à-dire qu'on partage l'ensemble d'apprentissage en 10 ensembles par exemple. Puis, on entraîne avec 9 d'entre eux en testant sur celui qu'on a laissé de côté. On répète l'opération en permutant l'ensemble de test et on fait une moyenne. C'est ce qui assure l'un des meilleurs conditionnement d'un point de vue statistique lorsqu'on a peu d'exemples. La stratification permet d'assurer un équilibre supplémentaire entre les différentes classes dans chacun des sous-ensembles. Une autre possibilité si on a suffisamment d'exemples est de partager avec un certain pourcentage (66 % ici par défaut) l'ensemble d'apprentissage en conservant 34 % pour l'ensemble de test.

Ensuite, on présente des chiffres de bases : taux de classification correcte global par exemple. Ici 33%, ce qui est peu mais correspond au choix basique de l'algorithme. Et constitue une base de comparaison pour les autres algorithmes.

Puis, on présente des chiffres plus spécifiques. **TP Rate** pour taux de Vrais Positifs (True Positive) ou taux de reconnaissance. **FP Rate** pour le taux de Faux Positifs (False Positive) ou taux de Fausses Alarmes.

La précision définie comme TP/(TP+FP) et le *Recall* ou Rappel définie comme TP/(TP+FN) où FN est le nombre de Faux Négatifs.

A quoi correspondent les Faux Négatifs ?

Souvent, on présente une courbe *Recall* contre Précision car ces deux valeurs évoluent en fonction de certains paramètres des algorithmes. Selon vous, quelle est la meilleure région dans ce genre de courbe ?

Interprétez les chiffres proposés pour l'algorithme utilisé !

Enfin, on présente une matrice dite Matrice de Confusion. Elle se lit ainsi : 50 éléments ont été classés *Iris-setosa* et sont des *Iris-setosa* ; 50 autres ont été classés *Iris-setosa* alors qu'ils sont des *Iris-versicolor* et de même pour les *Iris-virginica*.

Analysez les résultats obtenus si on utilise l'algorithme IB1 qui correspond à un algorithme du type 1-ppv ou 1-NN (1 plus proche voisin ou 1 Nearest Neighbour). Répétez avec un 2-NN.

Analysez les résultats obtenus si on utilise l'algorithme J48 qui est une version améliorée des arbres de décision type ID3. Représentez graphiquement sur papier l'arbre proposé et de fait écrivez les règles SI…ALORS qui en résultent.

Répétez avec l'algorithme J48-PART qui fournit directement une liste de règles.

Répétez avec un classifieur Bayésien Naïf.

Le panneau Cluster

Maintenant, vous allez pouvoir tester par vous-même dans ce panneau l'algorithme des *K-Means*. Une fois lancé l'algorithme (en ayant pris soin de cocher l'évaluation *Clusters vs. Classes)* et les résultats obtenus, dans la zone *Result List* cliquez avec le bouton droit de la souris sur l'algorithme courant et demandez de visualiser le regroupement. En utilisant le bon couple d'attributs, vous verrez les croix correspondant aux éléments correctement classifiés et les rectangles aux éléments incorrectement classifiés.

N.B. : un cluster est finalement une classe trouvé *a priori* (sans apprentissage) et la classe correspond à la classe réelle étiquetée par un superviseur.

Envol et Etude du temps.

Certains algorithmes ne fonctionnent qu'avec des données nominales.

Ouvrez le fichier *weather.nominal.arff*. Etudiez-le.

Appliquez successivement les algorithmes ID3 ou J48 (dans le panneau *Classify*) et APRIORI (dans le panneau *Associate*) et analysez les résultats.

Explorez par vous-même d'autres solutions proposées (5 minutes). A votre avis, quelle est la meilleure stratégie pour interpréter des données ?

Ouvrez à présent le fichier *weather.arff* qui contient un mélange de données numériques et nominales. Essayez les algorithmes précédents.

La solution est d'appliquer des filtres pour transformer les données. Par exemple, pour passer de variables numériques à nominales, on utilise le filtre *DiscretizeFilter*. Il faut l'ajouter à la liste des filtres puis *Apply Filter*. Enfin, Replace les données pour agir sur les données modifiées. Observez attentivement les retours dans les zones d'informations.

Essayez à nouveau les algorithmes précédents. Puis supprimer le filtre de la liste d'attente.

Annexe :

Des liens intéressants/importants :
http://javatechniques.com/blog/launching-java-webstart-from-the-command-line/
http://wiki.linux-france.org/wiki/Les_commandes_fondamentales_de_Linux