Fouille de Données - TP1 : Weka, règles d'association, clustering, classification

M2 – TIW2 – 2015/2016

Cette séance pour but de prendre en main **weka***, une plateforme d'algorithmes de data mining écrite en java que nous réutiliserons ainsi que d'expérimenter l'algorithme APriori de génération de règles d'association et les algorithmes de clustering et de classification vus en cours.*
Un compte rendu n'est pas demandé, toutefois il est fortement conseillé d'avoir fini ce TP avant le prochain. Vous pouvez également travailler en groupes de 4 personnes maximum.

Pour la deuxième séance, **veillez à télécharger et installer**[1] KNIME **(**`https://www.knime.org/`**).**

1 Présentation et installation de Weka

Weka est un ensemble de classes et d'algorithmes en Java implémentant les principaux algorithmes de data mining. Il est disponible gratuitement à l'adresse `www.cs.waikato.ac.nz/ml/weka`, dans des versions pour Unix et Windows. Ce logiciel est développé en parallèle avec un livre : Data Mining par I. Witten et E. Frank (éditions Morgan Kaufmann). Weka peut s'utiliser de plusieurs façons :

- Par l'intermédiaire d'une interface utilisateur : c'est la méthode utilisée dans ce TP.
- Sur la ligne de commande.
- Par l'utilisation des classes fournies à l'intérieur de programmes Java : toutes les classes sont documentées dans les règles de l'art. Nous y reviendrons sans doute dans un prochain TP.

1. Téléchargez Weka et installez le.
2. Téléchargez l'excellente présentation d'Eibe Frank à `http://liris.cnrs.fr/marc.plantevit/ENS/TP/weka.ppt` et parcourez là (un tutorial sur Weka est également disponible : `http://liris.cnrs.fr/marc.plantevit/ENS/TP/Tutorial.pdf`).

2 Premiers pas

Weka est maintenant installé sur votre compte. Après l'avoir lancé, vous obtenez la fenêtre intitulée Weka GUI Chooser : choisissez l'Explorer. La nouvelle fenêtre qui s'ouvre alors (Weka Knowledge Explorer) présente six onglets :

Preprocess : pour choisir un fichier, inspecter et préparer les données.

Classify : pour choisir, appliquer et tester différents algorithmes de classification : là, il s'agit d'algorithmes de classification supervisée.

Cluster : pour choisir, appliquer et tester les algorithmes de segmentation.

Associate : pour appliquer l'algorithme de génération de règles d'association.

Select Attributes : pour choisir les attributs les plus prometteurs.

Visualize : pour afficher (en deux dimensions) certains attributs en fonctions d'autres.

3 Les données

Les données sont sous un format ARFF -pour Attribute-Relation File Format-. Des exemples de données sont disponibles une fois weka installé[2]. Ouvrez dans un éditeur un de ces fichiers d'exemples et regardez son format.

1. avec toutes ses extensions possibles.
2. Au cas où, `http://liris.cnrs.fr/marc.plantevit/ENS/TP/data/`

Il est simple et il est facile de convertir des données-par exemple issues d'un tableur- en ARFF. (Il y a même un convertisseur inclus dans Weka du format csv vers le format arff).

Dans l'onglet Preprocess, cliquez sur Open File et ouvrez par exemple le fichier iris.arff : il contient la description de 150 spécimens d'iris de trois sortes différentes. Chaque description est composée de quatre attributs numériques (dimensions des sépales et des pétales), et d'un cinquième attribut qui est la classe de cet exemple (i.e. la sorte d'iris à laquelle il appartient). Pour chacun des attributs, vous pouvez obtenir, en cliquant dessus dans la sous-fenêtre Attributes, des statistiques basiques sur la répartition des valeurs pour cet attribut (sous-fenêtre Selected Attribute). On peut appliquer différents filtres aux données ; nous y reviendrons tout à l'heure.

4 Visualisation des données

Pour une première approche des données, passez dans la fenêtre Visualize. Vous y voyez un ensemble de 25 graphiques (que vous pouvez ouvrir en cliquant dessus), qui représentent chacun une vue sur l'ensemble d'exemples selon deux dimensions possibles, la couleur des points étant leur classe. Sur le graphique, chaque point représente un exemple : on peut obtenir le descriptif de cet exemple en cliquant dessus. La couleur d'un point correspond à sa classe (détaillé dans la sous-fenêtre Class colour). Au départ, le graphique n'est pas très utile, car les axes représentent le numéro de l'exemple.

1. Changez les axes pour mettre la largeur des pétales en abscisse, et la longueur des sépales en ordonnées.
2. Proposez un ensemble de deux règles simples permettant de classer les exemples selon leur genre : quelle erreur commettrez-vous ? Les petits rectangles sur la droite de la fenêtre représentent la distribution des exemples, pour l'attribut correspondant, par rapport à l'attribut (ou la classe) codé par la couleur. En cliquant du bouton gauche sur un de ces rectangles, vous le choisissez comme axe des X, le bouton droit le met sur l'axe des Y.
3. En mettant la classe sur l'axe des X, quels sont à votre avis les attributs qui, pris seuls, permettent le mieux de discriminer les exemples ? Si les points sont trop serrés, le potentiomètre Jitter, qui affiche les points "à peu près" à leur place, vous permet de les visualiser un peu plus séparément : cela peut être utile si beaucoup de points se retrouvent au même endroit du plan.

5 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 ?

5.1 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 que L désigne la partie gauche de la règle et R la partie droite. Il y en a quatre (L désigne la partie gauche de la règle et R la partie droite) :
- Confidence : la confiance.
- Lift : l'amélioration.
- 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 :
- Conviction : similaire à l'amélioration, mais on s'intéresse aux exemples où la partie droite de la règle n'est pas respectée. Le rapport est inversé.

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, jouer avec les paramètres.

6 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 :

6.1 Transformation des données

Récupérer le fichier bank-data.csv [3] Revenez à la fenêtre Preprocess.

1. Tout d'abord ouvrez le fichier bank-data.csv : il vous sera proposé d'utiliser un convertisseur : dites-oui ! 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 sous-fenêtre Filters vous permet de manipuler les filtres. Le fonctionnement général est toujours le même :
   — Vous choisissez un ensemble de filtres, chaque filtre, avec ces options, étant choisi dans le menu déroulant du haut de la sous-fenêtre, puis ajouté à la liste des filtres par la commande Add.
   — On applique les filtres avec la commande Apply Filters.
   — On peut alors remplacer le fichier précedemment chargé par les données transformées, à l'aide du bouton Replace.
   — Ce fichier devient alors le fichier de travail.
   — Le bouton Save sauvegarde ces données transformées dans un fichier.
2. Pouvez vous lancer l'algorithme Apriori ? Pourquoi ?

6.2 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 !

6.3 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...

3. http ://archive.ics.uci.edu/ml/

6.4 APriori

Sauvez dans bankd.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 ?

6.5 Classification avec Apriori

Reprenez le fichier iris.arff ; discrétisez les attributs continus, vous pouvez également créer de nouveaux attributs qui peuvent la combinaison d'autres (par exemple, $(A_1 \times A_2)^2$). Appliquez ensuite l'algorithme Apriori. Examinez les règles produites avec comme conclusion uniquement la classe : correspondent-elles à votre intuition ? Comparez avec ce qui est produit par un algorithme de classification, en choisissant par exemple dans trees, J48 qui construit un arbre de décision.

7 Promotions de Noël et épicerie de nuit

L'épicerie de nuit de la rue *"remplacez par votre rue favorite"* a décidé à l'approche des fêtes de fin d'année 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 2006-2007

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 ?

8 Mise en œuvre

Rendez vous sur le site `http://archive.ics.uci.edu/ml/`, choisissez un jeu de données et importer le dans weka afin de le visualiser et extraire des règles d'association.

9 API Weka

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).

Clustering – Introduction

Nous allons utiliser le paquetage *weka.clusterers* pour l ?analyse de données qui ne contiennent pas d'attribut de classe. Ainsi, puisque presque tous les ensembles de données que nous utilisons possèdent un attribut de classe, nous allons ignorer ces attributs (sauf pour les phases d ?évaluation). Nous allons utiliser le **Weka Knowledge Explorer**.

Remarque : Si vous souhaitez voir les commandes spécifiques des algorithmes, vous pouvez utiliser le simple CLI (command line options) : `java weka.clusterers.Cobweb -h` pour le clustering conceptuel. Ou encore : `java weka.clusterers.SimpleKMeans -h` pour le k-means clustering.

10 Premiers contacts

Les classes qui implémentent une méthode de clustering dans l'outil Weka, sont regroupées dans le package weka.clusterers. La classe weka.clusterers.Clusterer définit la structure générale commune à toutes les méthodes de clustering. A partir de l'onglet Cluster on peut observer quatre algorithmes implémentés : **SimpleKMeans**, **EM** (expectation-maximization), **CobWeb** et **FarthestFirst**.

Weka affiche le nombre d'exemples assignés à chaque cluster. Weka permet de tester la qualité du modèle sur un jeu de test. C'est la mesure de vraisemblance (log-likelihood) qui est utilisée. Plus la mesure est grande, mieux le modèle caractérise les données. Le test peut être effectué par validation croisée.

La boîte Cluster mode permet de choisir la méthode d'évaluation du modèle extrait.

Use training set : effectue et teste le clustering sur le même jeu de données ;

Supplied test set : teste le clustering sur un jeu de données à spécifier ;

Percentage split : effectue le clustering sur le pourcentage indiqué du jeu de données et teste sur le pourcentage restant ;

Classes to clusters evaluation : teste le clustering relativement à une classe

Pour cet exercice, on considérera les données du fichier **vote.arff**. Ce jeu de données décrit le résultat des votes de chaque représentant au Congrès des Etats-Unis sur les 9 questions clés identifiés par le Congressional Quarterly Almanac.

10.1 Analyse exploratoire

Effectuez une première analyse du jeu de données. Combien d'instances ? Combien d'attributs ? Quels types ? etc.

10.2 K-Means

Effectuez un clustering du jeu de données en utilisant l'algorithme **SimpleKMeans** et en conservant les paramètres par défaut.

Le résultat du clustering est donné avec une instance par cluster représentant le centroïde du cluster.

```
kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 1510.0
Missing values globally replaced with mean/mode

Cluster centroids:
                                              Cluster#
Attribute                               Full Data          0          1
                                          (435)      (214)      (221)
=================================================================================
handicapped-infants                            n          n          y
water-project-cost-sharing                     y          y          n
adoption-of-the-budget-resolution              y          n          y
physician-fee-freeze                           n          y          n
el-salvador-aid                                y          y          n
religious-groups-in-schools                    y          y          n
anti-satellite-test-ban                        y          n          y
aid-to-nicaraguan-contras                      y          n          y
mx-missile                                     y          n          y
immigration                                    y          y          y
synfuels-corporation-cutback                   n          n          n
education-spending                             n          y          n
superfund-right-to-sue                         y          y          n
crime                                          y          y          n
duty-free-exports                              n          n          y
export-administration-act-south-africa         y          y          y
Class                                 democrat republican   democrat


Clustered Instances

0       214 ( 49%)
1       221 ( 51%)
```

Visualisation :

A partir de la boîte **Result List** (bouton droit, **Visualize cluster assignement**), visualisez la répartition des exemples dans chaque cluster.

Évaluation relativement à une classe

L'option d'évaluation **Classes to clusters evaluation** permet d'assigner une classe à un cluster pendant la phase de test. La classe assignée est la plus fréquente dans le cluster ; une erreur de classement (taux de mal classés) est calculée ainsi que la matrice de confusion. Dans ce cas, l'algorithme ne prend pas en compte la valeur de cet attribut dans le calcul de distance.

- Effectuez un clustering du jeu de données en utilisant la méthode implémentée par SimpleKMeans en conservant les paramètres par défaut avec l'option Classes to clusters evaluation et l'attribut de classe class
- Recommencez en modifiant l'attribut de classe et observer les taux d'erreur
- Conservez les meilleurs résultats et effacer les autres de la liste des résultats. Quel est l'attribut de classe pour lequel l'erreur est la plus faible ?
- Visualisez : A partir de la boîte Result List (bouton droit, Visualize cluster assignement), visualiser la répartition des exemples dans chaque cluster. Les croix représentent les instances classées dans le "bon" cluster et les carrés représentent les instances classées dans le "mauvais" cluster.
- Exportez le résultat de votre clustering (le meilleur) et caractérisez les clusters via des règles d'association.

Comparaison de modèles

- Lancez l'algorithme SimpleKMeans plusieurs fois avec les valeurs 20, 50, 100, 1000 pour le paramètre random seed avec l'option **Classes to clusters evaluation** et l'attribut de classe **class** et en fixant le nombre de clusters à 2.
- Quel est le meilleur résultat selon le taux d ?erreur et la taille de clusters.
- Conservez le meilleur résultat

10.3 EM

La méthode EM (Expectation Maximisation) génère une description probabiliste des clusters en terme de moyenne et écart-type pour les attributs numériques et en terme de nombre pour les attributs nominaux. Chaque cluster est décrit par sa probabilité a priori et une distribution de probabilité pour chaque attribut. Pour un attribut nominal, est affiché le nombre d'exemples et pour un attribut numérique est affiché les caractéristiques de sa distribution normale. L'option d'évaluation **Classes to clusters evaluation** affiche aussi le **log-likelihood**, (ou log-vraissemblance) assigne une classe au cluster, calcule l'erreur et la matrice de confusion.

— Effectuez un clustering du jeu de données en utilisant la méthode EM avec les paramètres par défaut. Combien de classes sont découvertes ?

Évaluation relativement à une classe

— Effectuez un clustering du jeu de données en utilisant la méthode EM en fixant le nombre de clusters à 2 et avec l'option **Classes to clusters evaluation** et l'attribut de classe **class**.
— Effectuez un clustering du jeu de données en utilisant la méthode EM en fixant le nombre de clusters à 2 et avec l'option **Classes to clusters evaluation** et l'attribut de classe **el-salvador-aid**.
— Comparez les résultats à ceux obtenus à la question **??**.

10.4 Clustering hiérarchiques avec Cobweb

La méthode Cobweb réalise un clustering hiérarchique où les clusters sont décrits de manière probabiliste. L'algorithme possède deux options importantes : **Cutoff** (par défaut=0.002) et **Acuity** (par défaut=1.0). ? ?

— Lancez un clustering du jeu de données en utilisant la méthode Cobweb avec les paramètres par défaut et avec l'option Classes to clusters evaluation et l'attribut de classe class La fenêtre d ?output montre le nombre de noeuds regroupés puis découpés, le nombre de clusters et la structure hiérarchique de clusters. Quel est le nombre de clusters trouvés ?

Évaluation relativement à une classe

— Fixez le paramètre Cutoff à 0.5 de manière à supprimer le découpage (split) de noeuds et donc ramener le nombre de clusters à 2 et avec l'option Classes to clusters evaluation et l'attribut de classe class
— Conservez le paramètre Cutoff à 0.5 avec l'option Classes to clusters evaluation et l'attribut de classe el-salvador-aid
— Comparez les résultats obtenus avec les trois méthodes de clustering.

11 Jeux de données de base pour comparer les techniques de clustering

11.1 Jeu de données « ligne-carré »

En utilisant le jeu de données à 2 attributs x et y pour la représentation des clusters lignes-carrés (lignec1.arff et lignec2.arff),

— Testez l ?algorithme des K-moyennes et celui l ?Expectation-Maximization.
— Permettent-ils de retrouver les groupes identifiables graphiquement ?
— Visualisez les assignements aux clusters.

11.2 Jeu de données par distribution sur chaque attribut

En utilisant les jeux de données pour la distribution (em2.arff et em3.arff).

— Testez l ?algorithme des K-moyennes et l ?Expectation maximization.
— Permettent-ils de retrouver les groupes identifiables graphiquement ?
— Visualisez les assignements aux clusters.

11.3 Données Titanic

— Filtrez les données "titanic.arff" ("titanic.txt") pour enlever la prédiction (SURVIVED) des données d ?apprentissage.
— Les algorithmes de clustering des K-moyennes et l ?Expectation maximization permettent-ils de découper les données en un groupe de survivants et un groupe de non-survivants ?
— Visualisez les assignements aux clusters.

11.4 Iris

— Testez les algorithmes des K-moyennes et l ?Expectation maximization, en faisant varier les paramètres sur le jeu de données Iris.
— Ignorez des attributs et tester l ?influence sur le résultat.
— Visualisez les assignements aux clusters.

11.5 Bilan

— Testez si possible d'autres jeux de données (de préférence les plus grands i.d. soybean, labour, etc.) et analysez les différents clusters fournis avec SimpleKMeans, EM et Cobweb.
— Finalement, listez les principaux avantages et désavantages de ces algorithmes de clustering utilisés durant la séance.
— Evaluer la qualité d ?un clustering est toujours difficile lorsque l ?on compare différentes exécutions. D ?après ce que vous avez constaté durant cette séance, quels critères pourriez-vous employer pour la qualite ? des clusters.

12 Application

Cet exercice porte sur les jeux de données Clients (Clients.csv) et Immatriculations (Immatriculations.csv) Voici quelques informations sur les attributs

Pour Clients.csv : ?
— taux : exprime en euros la capacité d'endettement du client (correspond à environ 30% de son salaire)
— 2eme voiture : valeur booléenne indiquant si le client possédait déjà un véhicule principal

Pour Immatriculations.csv :
— puissance : exprimée en chevaux
— longueur : quatre catégories ont été déterminées « courte », « moyenne », « longue » et « très longue »
— prix : en euros

On demande de lancer différentes méthodes de clustering sur ces deux jeux de données et de présenter les solutions les plus pertinentes extraites (on pourra considérer le jeu de données Clients_1.csv dans lequel l'attribut immatriculation a été supprimé). Il s'agira de sélectionner différents attributs des jeux de données initiaux et de lancer différents algorithmes de clustering en faisant varier la valeurs de leurs paramètres

13 Utilisation de weka sans l'interface

Dans cette partie, on va utiliser les classes Java de Weka (voir la doc en ligne).
— Vous avez dû constater que l'algorithme K-Means est sensible à l'initialisation. Écrivez un programme Java qui prend en argument un fichier de données et applique l'algorithme avec différentes graines, et retourne pour chaque valeur l'erreur "SquaredError". Vous pouvez retournez la meilleure segmentation.
— Un autre inconvénient de l'algorithme K-Means est de devoir fixer le nombre de clusters. Écrivez un programme qui prend en argument un fichier de données et qui applique l'algorithme avec différents nombres de clusters. Pour chaque valeur, vous pourrez appliquer l'algorithme avec différentes graines et prendre l'erreur moyenne. Affichez les différentes erreurs moyennes pour chaque valeur de nombre de clusters (k). Testez sur les exemples précédents. Que constatez vous ? Cette approche peut-elle vous aider à trouver le nombre « idéal » de clusters pour le jeu de données considéré ?

Remarque : pour interpréter plus facilement les résultats, on peut visualiser le graphe de la fonction qui associe l'erreur moyenne au nombre de clusters. Pour cela, vous pouvez utiliser **gnuplot**. *Il suffit de créer un fichier contenant les valeurs (une ligne par point de la forme x, y et sous gnuplot de l'afficher avec la commande* `plot nomfichier with lines`. *Pour ceux qui sont sous Windows, il vous reste les outils de bureautique habituels.*

14 Détails sur l'évaluation d'une classification

Pour effectuer une classification, différents points doivent être pris en compte :
— Les effets des jeux de tests utilisés ;
— Les effets du type de validation de modèles :
  — Validation sur le jeu de données ayant servi à l'apprentissage ;
  — Validation sur un autre jeu de données (sur une deuxième partition des données) ;
  — Validation croisée.

Nous allons au cours de cette séance et de la suivante voir l'influence de ces deux éléments sur les résultats de la classification et sur la qualité des solutions obtenues (qualité réelle et mesures fournies par les indicateurs).

A la fin de cette séance sur le **Weka Experiment Environment**, vous serez capables de concevoir des expériences pour évaluer les schémas des classifieurs sur de multiple jeux de données.

15 Introduction Weka Experimenter

Le *Weka Experiment Environment* permet à l'utilisateur de créer, lancer, modifier et analyser des expériences de manière plus souple que l'utilisation des classifieurs individuellement. Par exemple, l'utilisateur peut créer une expérience qui lance plusieurs classifieurs sur des séries de jeux de données et analyse les résultats pour déterminer si l'un des classifieur est statistiquement meilleur que les autres. Nous allons utiliser l'interface de l'Experiment Environment puisqu'il fournit plus de flexibilité à l'utilisateur pour développer des expériences que cela est possible en tapant les commandes dans une simple console (CLI). Pour commencer avec l'interface de l'Experiment Environment, démarrez Weka et cliquez sur *Experimenter* dans la fenêtre *Weka GUI Chooser*.

15.1 Définir une expérience

Quand l'*Experimenter* est démarré, la fenêtre Setup est affichée. La première étape consiste à choisir un jeu de données de travail.

1. Cliquez sur New (en haut à droite de la fenêtre Setup) pour initialiser une expérience. Ceci permet de fixer les paramètres par défaut pour l'expérience. Pour définir le jeu de données (dataset) à traiter par un classifieur, sélectionnez d'abord *Use relative paths* dans la fenêtre *Datasets* (en bas à gauche de la fenêtre Setup) et cliquez sur *Add new* pour ouvrir une fenêtre de dialogue. Double-cliquez sur data pour voir les datasets disponibles ou naviguez vers un autre répertoire. Sélectionnez le dataset Iris. Le nom du dataset est maintenant affiché dans la fenêtre Datasets de la fenêtre de Setup.

Se placer en mode *advanced* en haut de la fenêtre.

15.2 Sauvegarder les résultats

Afin de ne pas refaire le même travail plusieurs fois et de pouvoir exploiter les résultats de l'expérience, on peut sauvegarder les résultats de l'expérience dans un fichier.

1. Pour identifier un fichier vers lequel les résultats seront envoyés, cliquez sur *CSVResultListener* dans la fenêtre Destination (en haut de la fenêtre Setup). Se placer sur le paramètre de l'output *outputFile*. Cliquez sur ce paramètre pour afficher une fenêtre de sélection de fichiers. Tapez le nom du fichier de sortie (output), par exemple Experiment.txt ou le sélectionner via open, cliquez sur Select, et cliquez sur close (X). Vérifiez que tous les fichiers de sortie sont situés dans votre répertoire (et pas dans le répertoire weka). Le nom du fichier est affiché dans la fenêtre outputFile. Cliquez sur OK pour fermer la fenêtre. Le nom du dataset est affiché dans la fenêtre Destination du Setup.

15.3 Sauvegarder la définition de l'expérience

La définition d'une expérience peut être sauvegardée à tout moment.

1. Sélectionnez Save en haut de la fenêtre Setup. Tapez le nom du dataset avec l'extension "exp" (e.g. Experiment.exp).

Récupération : L'expérience peut être récupérer en sélectionnant Open(en haut à gauche de la fenêtre Setup) et sélectionnez Experiment.exp dans la fenêtre de dialogue.

Pensez à sauvegarder les définitions de vos expériences de manière à pouvoir les réutiliser plus tard.

15.4 Lancer une expérience

Sans autre indication, c'est l'expérience par défaut est exécutée sur le dataset que vous avez choisi, soit, normalement le CrossValidationResultProducer avec un paramètre à 10 pour l'apprentissage aléatoire et des tests effectués sur le dataset choisi (Iris), utilisant le classifieur ZeroR.

Remarque : Il faut se placer en mode disabled dans le generator properties.

1. Pour lancer l'expérience que vous venez de configurer, cliquez sur l'onglet Run en haut de la fenêtre Experiment Environment et cliquez sur Start. L'expérience est exécutée.

```
Started
Finished
There were 0 errors
```

Si l'expérience a été définie correctement, les 3 messages présentés ci-dessus sont affichés dans le Log Panel. Les résultats de l'expérience sont sauvegardés dans le fichier Experiment.txt. C'est une fichier de type .CSV (avec valeurs séparées par des ","). Il faut effectuer quelques modifications grâce à un éditeur (de type textpad), et remplacer les ',' par des ' ;', puis les '.' (séparateur de décimales en anglais qui serait évalué comme séparateur de millier par un tableur) par des ','. Ensuite le fichier peut être chargé dans un classeur (Excel ou OpenOffice) pour être analysé.

1. Chargez ce fichier dans une feuille Excel. Spécifiez le délimiteur à " ;".

La ligne 1 contient les identifiants de colonnes (du résultat de l'expérience). Chaque ligne définit un jeu d'apprentissage et un test. La ligne 2 de la feuille de données indique que pour le premier run de l'expérience, le dataset Iris a été utilisé avec le classifieur ZeroR et que W instances ont été testées par le classifieur : X instances ont été classées correctement, Y instances ont été mal classées, et Z instances n'ont pas pu être classées.

1. Laquelle (lesquelles) de ces colonnes pourrai(en)t être utile pour analyser l'efficacité d'un classifieur ?

15.5 Modifier les paramètres de l'expérience

Les paramètres d'une expérience peuvent être modifiés en cliquant sur la partie Result Generator (sous Destination dans la fenêtre de Setup). Redimensionnez la fenêtre pour avoir tous les paramètres visibles. Le RandomSplitResultProducer réalise des apprentissages et tests de manière répétée. Le nombre de cas (ex- primé sous forme de pourcentage) à utiliser pour l'apprentissage est donné dans le champ trainPercent. Le nombre de runs est spécifié dans la fenêtre Setup dans le paramètre Runs. Un petit fichier d'aide peut être affiché en cliquant More dans le panneau About.

1. Cliquez sur l'entrée splitEvaluator pour afficher les propriétés du SplitEvaluator. Cliquez sur l'entrée classifier (ZeroR) pour afficher les propriétés du classifieur. Ce classifieur n'a pas de propriétés modifiables mais la plupart des autres classifieurs en ont (e.g. j48.J48) et peuvent être modifiés par l'utilisateur. Cliquez sur la liste déroulante pour le classifieur ZeroR (choose) et changez le en j48.J48 pour le classifieur arbre de décision.
2. Vous pouvez modifier les paramètres, par exemple augmentez le minNumObj de 2 à 5 (i.e. spécifiez le nombre minimum de cas dans les nœuds feuilles).
3. Cliquez maintenant sur OK pour fermer la fenêtre. Le nom du nouveau classifieur est affiché dans le panneau Result generator. Vous pouvez relancer l'expérience. Le fichier d'output (Experiment.txt) sera écrasé par les résultats du j48.J48.

15.6 Comparer des classifieurs

Pour comparer plusieurs classifieurs, nous devons les ajouter dans le panneau de Generator properties.

1. Pour commencer, modifiez l'entrée drop-down de Disabled à Enabled dans le panneau Generator properties (en bas à droite de la fenêtre Setup). Cliquez sur Select property et sélectionnez splitEvaluator de manière à ce que l'entrée classifier soit visible dans la liste des propriétés. Cliquez sur Select. Le nom du classifieur est affiché dans le panneau Generator properties. Sélectionnez ZeroR comme classifieur. Puis utilisez le bouton Add pour l'ajouter à l'expérience. Pour ajouter un autre classifieur, cliquez sur le nom du classifieur pour afficher sa fenêtre de propriétés. Cliquez sur la liste déroulante et sélectionnez j48.J48 , le classifieur arbre de décision. Le nouveau classifieur est ajouté dans le panneau Generator properties. Cliquez sur Add pour l'ajouter. Lancez l'expérience et regardez le fichier résultat Experiment.txt. Vous verrez que les résultats ont été générés à partir des deux classifieurs. Pour ajouter d'autres classifieurs, répétez le procédé. Pour enlever un classifieur, sélectionnez le en cliquant dessus et cliquez Delete.

15.7 Ajouter des jeux de données

Le(s) classifieur(s) peuvent être lancés sur un nombre quelconque de datasets à la fois. Les datasets sont ajoutés en cliquant sur Add new dans le panneau Datasets (en bas à gauche de la fenêtre Setup). Les datasets sont effacés de l'expérience en sélectionnant le dataset voulu et en cliquant sur Delete Selected.

16 Les différentes méthodes de validation de modèles

Il existe différentes méthodes de validation de modèles en Weka :

— Supplied test set (avec paramètre set - choix du jeu de données pour la validation) : consiste à évaluer le modèle sur un autre jeu de données (a priori différent de celui utilisé pour construire le modèle)

— Cross-validation (avec paramètre folds) : consiste à diviser les données en n groupes. On construit les modèles sur n-1 groupes et on les teste sur le nième groupe. Puis on change de groupe test et on répète le même procédé jusqu'à avoir réalisé toutes les combinaisons. On considère alors la moyenne des validations comme la validation finale.
— Percentage split (avec paramètre pourcentage) : consiste à utiliser un certain pourcentage des données pour construire le modèle et l'autre partie pour le valider.

17 Analyser les résultats avec Weka

En plus d'envoyer les résultats d'une expérience dans un CSV Result Listener (fichier d'output), ces résultats peuvent également être envoyés à un InstancesResultListener et analyser par le Weka Experiment Analyser. Ceci permettra de ne pas utiliser un logiciel tel qu'excel (cf exercice précédent) pour l'analyse des résultats.

1. Cliquez sur la partie Result Listener du panneau Destination et sélectionnez InstancesResultListener. N'oubliez pas de vous placer en mode advanced (en haut de la fenêtre). Spécifiez le nom du result output. L'output sera sous un format dataset, donc spécifiez l'extension "arff" (e.g. Experiment4.arff).

Une fois encore assurez-vous que les fichiers d'output sont enregistrés dans votre répertoire personnel. L'output créé avec le InstancesResultListener est dans un format "arff" de type :

```
   @relation InstanceResultListener
@attribute Key_Dataset {iris}
@attribute Key_Run {1,2,3,4,5,6,7,8,9,10}
@attribute Key_Scheme {weka.classifiers.ZeroR}
@attribute Key_Scheme_options {''}
@attribute Key_Scheme_version_ID {6077547173920530258}
@attribute Date_time numeric
@attribute Number_of_instances numeric
@attribute Number_correct numeric
@attribute Number_incorrect numeric
@attribute Number_unclassified numeric
@attribute Percent_correct numeric...
@data
```

Chacunes des instances de ce dataset représente les informations relatives à un run (une ligne dans les tables excel utilisées précédemment).

Le Weka Experiment Analyzer peut maintenant être utilisé pour effectuer le travail d'analyse des résultats des expériences (envoyés à un InstancesResultListener).

18 Expériences

18.1 Premier pas

1. Définissez une expérience respectant les règles suivantes :
   (a) appliquant 10 runs répétitifs d'apprentissage /test
   (b) utilisant les datasets Iris et Soybean
   (c) avec ZeroR, OneR et j48.J48

   Une fois l'expérience entièrement configurée, lancez celle-ci.
2. Pour analyser les résultats, sélectionnez l'onglet Analyse en haut de la fenêtre de l'Experiment Environment. (Notez que les résultats doivent être sous le format arff. Vous pouvez visualiser le fichier d'output généré avec un éditeur de texte standard.) Cliquez sur Experiment pour analyser les résultats de l'expérience courante. Le nombre de lignes de résultat disponibles ("Got 30 results") est affiché dans le panneau Source. Cette expérience est composée de 10 runs, pour 3 classifieurs, pour 1 dataset, soit un total de 30 lignes de résultats.
3. Quelles conclusions peuvent être déduites à propos des performances des classifieurs sur les différents datasets ?
4. En appuyant sur le bouton Select Base, sélectionnez ZeroR comme étant le classifieur de base. Tous les classifieurs seront comparés à ce classifieur de base. Sélectionnez l'attribut $Percent_{correct}$ du champ Comparison et cliquez sur Perform test pour générer une comparaison des 3 classifieurs.

Comprendre : Il existe une colonne pour chacun des classifieurs utilisés dans l'expérience, et une ligne pour chacun des datasets utilisés. Le pourcentage de "correction" pour chaque classifieur est affiché pour chaque ligne dataset : e.g. X% pour ZeroR, Y% pour OneR, et Z% pour j48.J48. L'annotation "v" ou "*" indique qu'un résultat spécifique est statistiquement meilleur (v) ou pire (*) que le classifieur de base (dans le cas présent, ZeroR) avec un niveau de signification précisé (ici 0.05).

Les résultats de OneR et j48.J48 devraient normalement être nettement meilleurs que la base de référence établi par ZeroR. En bas de chaque colonne (sauf pour la première) on trouve un compteur (xx/ yy/ zz) du nombre de lignes pour lequel le classifieur a été meilleur que (xx), le même que (yy), ou pire que (zz) le classifieur de base sur le dataset utilisé dans le run.

Dans l'exemple, il n'y a qu'un seul dataset et OneR a été 1 fois meilleur que ZeroR et jamais équivalent ou pire à ZeroR (1/0/0) ; j48.J48 est également meilleur que ZeroR sur le dataset. La valeur "(10)" au début des lignes "iris" précise le nombre de runs de l'expérience.

L'écart type de l'attribut évalué peut être calculé en sélectionnant la case std.deviations. En sélectionnant Number-correct en champ de comparaison et en cliquant sur Perform test, on géne ?re la moyenne du nombre de correct sur l'ensemble des tests (sur un maximum de 51 cas de test, 34% des 150 cas dans le dataset Iris).

18.2 Rang de test

1. Sélectionnez Ranking grâce au bouton Select base.

   Ceci fournit le nombre de classifieurs que chaque classifieur "surpasse" ou inversement par lesquels il est sur- passé. Le rang de test classe les classifieurs en fonction de leur nombre total de "victoires" ($>$) et de "défaites" ($<$) contre les autres classifieurs. La premiere colonne ($> - <$) fournit la différence entre le nombre de "victoires" et le nombre de "défaites".

18.3 Sauvegarder les résultats

Les informations affichées dans le panneau Test output (à droite de la fenêtre) sont controlées par l'entrée actuellement sélectionnée dans le panneau Result list (en bas à gauche de la fenêtre).

En cliquant sur l'une des entrées, les résultats correspondant à cette entrée sont affichés. Les résultats affichés dans le panneau Test output peuvent être sauvegardés dans un fichier en cliquant sur Save output.

Un seul jeu de résultats peut être sauvegardé à la fois, mais Weka permet à l'utilisateur de sauvegarder tous les résultats dans un même dataset en les sauvegardant un à la fois et en utilisant l'option Append au lieu de Overwrite pour les sauvegarder.

Le choix de cette option vous est offert via une fenêtre de dialogue apparaissant en milieu d'écran, après avoir sélectionné le fichier de sauvegarde.

19 Weka Analyser

19.1 Expériences avec la validation croisée (Cross-Validation)

1. Pour passer d'apprentissage et des tests aléatoires (comme dans l'exercice précédent) à des expériences par validation croisée, cliquez sur l'entrée Result generator. Cliquez sur la liste déroulante en haut de la fenêtre et sélectionnez CrossValidationResultProducer.

   La fenêtre contient maintenant les paramètres spécifiques pour la validation croisée tels que le nombres de partitions (folds). L'expérience par défaut travaille avec une validation croisée sur 10 partitions. Le panneau Result generator indique maintenant que la validation croisée sera utilisée. Vous pouvez cliquer sur More pour générer de brefs descriptions du producteur de résultats par validation croisée. Tout comme le RandomSplitResultProducer, de nombreux classifieurs peuvent être lancés pendant la validation croisée en les ajoutant au panneau Generator properties.

2. Définissez une expérience respectant les règles suivantes :
   — 10 runs de 10 partitions pour la validation croisée ;
   — utilisant 3 datasets (e.g. Iris, Labor, et Weather) ;
   — avec 3 méthodes différentes dont j48.J48.
   Faites en sorte de pouvoir analyser le fichier automatiquement.

19.2 AveragingResultProducer

Nous allons utiliser en alternative au CrossValidationResultProducer, le AveragingResultProducer.
Ce "producer" de résultats fournit la moyenne d'un jeu de tests (généralement des runs de validation croisée).

1. Cliquez sur le panneau Result Generator et sélectionnez AveragingResultProducer dans la liste déroulante.
   Tout comme pour les autres "producers", d'autres classifieurs peuvent être définis. Quand le AveragingResultProducer est utilisé, la propriété classifier est située plus loin dans la hiérarchie de Generator properties. Avec expectedResultsPerAverage fixé à 10, l'expérience consistera en 10 runs de 10 partitions de validation croisée. Chaque run de 10 partitions de validation croisée est alors moyenné, produisant une ligne de résultat pour chaque run (au lieu d'un résultat par ligne pour chaque partition dans l'exemple précédent utilisant le CrossValidationResultProducer).
2. Répétez l'exercice de la Section Cross-Validation en utilisant maintenant le AveragingResultProducer.
3. Pour chacun des classifieurs :
   — Effectuez une analyse de leur performance les uns par rapport aux autres.
4. Définissez une expérience pour comparer 3 classifieurs (IBK, J48 et OneR) par rapport au $Percent_{correct}$, sur les datasets Iris et Soybean. Souvenez-vous d'utiliser le InstancesResultListener et de sauvegarder les output dans un format arff (disons Experiment5.arff). Une fois les expériences effectuées, utilisez les utilitaires de Experiment Environement dans la partie Analyse. Pouvez-vous faire des commentaires de significations statistiques ? Vérifiez votre réponse en sélectionnant Summary comme base de Test et en appuyant sur Perform test. Ceci vous fournit une matrice résumant l'analyse de l'expérience.

   Comprendre : Comprendre : Par exemple, une ligne (- 1 1) indique que les classifieurs liés aux colonnes "b" et "c" sont meilleurs que celui lié à la ligne "a". Une entrée à 0 indique qu'il n'y a pas de différence significative entre le classifieur lié à la ligne et celui lié à la colonne.
5. Y-a-t-il des différences significatives par rapport au temps d'apprentissage et de test entre les classifieurs testés ? Comment trouver cette information ? Indice : Modifier la sélection du "comparison field".

20 Exploiter les résultats de façon automatique

Vous avez dû remarquer que les résultats des expériences sont stockés dans des fichiers arff. Rien ne vous empêche d'analyser ces résultats.

1. Appliquez une méthode de clustering sur les résultats de l'expérience.
2. Interprétez les regroupements.

21 Knime

Lors de la prochaine séance, nous utiliserons la plateforme KNIME qui subsume Weka, R et possède beaucoup plus de fonctionnalités que Weka. L'une de ses principales caractéristiques est la définition de workflow.
— **Veillez à télécharger et installer KNIME (`https://www.knime.org/`) avant la prochaine séance.**