Acquisition de connaissances
TP1 - Apprentissage supervisé

1 Avant-propos

Les données du TP sont issues des jeux de données fournis par l'*UCI (University of California, Irvine)* pour réaliser l'évaluation des méthodes d'apprentissage artificiel [1]. Elles sont disponibles au format `arff` (*Attribute-Relation File Format*) sur le site du cours à l'adresse suivante : `http://www.irisa.fr/texmex/people/claveau/cours/acqdeco2`

Les compte-rendus de TP sont à rendre au plus tard une semaine après la séance de TP par mail sous forme électronique (solution préférée), ou sous forme papier. N'hésitez pas à faire des captures d'écrans lorsque les questions s'y prêtent.

2 Aperçu de Weka

Weka est une archive jar qui se trouve dans /usr/local/stow. Depuis votre répertoire de TP, Weka peut se lancer par `java -jar /usr/local/stow/weka-3-5-6/weka.jar`

Weka possède quatre interfaces :
- `Simple CLI` : ligne de commande ;
- `Explorer` : interface d'exploration des données ;
- `Experimenter` : mise en place de jeux d'expériences ;
- `Knowledge Flow` : processus complet d'apprentissage, des données aux résultats ;

Nous utiliserons dans ce TP l'interface *Knowledge Flow* de Weka qui permet de bien appréhender les différentes étapes du processus d'apprentissage.

Rappel : un processus d'apprentissage comprend les étapes suivantes :

1. Analyse des données : identifier la quantité d'instances, le nombre et la nature des attributs, l'attribut que l'on souhaite prédire (classe). Éventuellement, sélectionner ou transformer les attributs.
2. Choix d'un classifieur : quel classifieur est le plus adapté en fonction du type et de la quantité de données ?
3. Choix de la stratégie d'évaluation : quelles sont les données d'apprentissage ? de test ? Quelles mesures d'évaluation utilise-t-on ? Faut-il prévoir des données de validation ?
4. Apprentissage : choix des paramètres, apprentissage sur l'ensemble d'évaluation et, le cas échéant, optimisation sur l'ensemble de validation.

1. `http://archive.ics.uci.edu/ml/`

5. Évaluation du classifieur sur l'ensemble de test : quelles sont les performances du classifieur obtenu ?
6. Visualisation des résultats : comment représenter les résultats obtenus ?

L'interface *Knowledge Flow* de Weka permet de modéliser ce processus sous forme d'un graphe dont chaque noeud correspond à un outil particulier (classifieur, filtrage d'attributs...) et dont les arêtes représentent la nature des données échangées entre deux outils (figure 1).

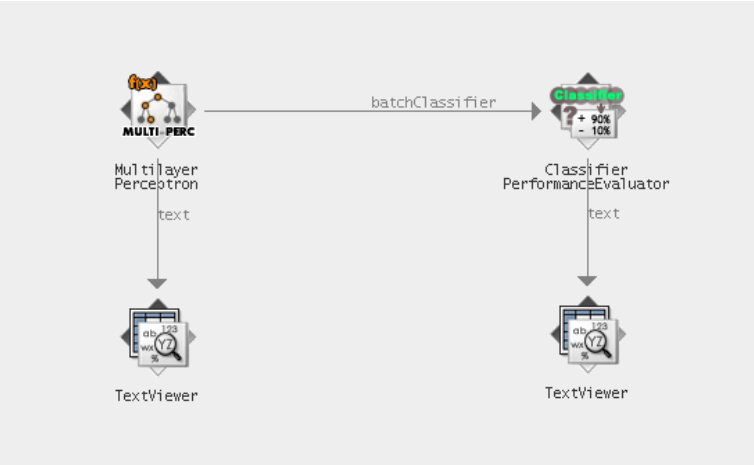

FIGURE 1 – Exemple classifieur-évaluation-visualisation : le classifieur est placé en entrée du module d'évaluation (arête *batchClassifier*), celui d'évaluation en entrée du module de visualisation (arête *text*). Le module de visualisation de gauche (resp. de droite) permet d'observer les résultats de l'apprentissage (resp. de l'évaluation) sous forme de texte.

Les différents outils disponibles sont classés par onglet, chaque onglet étant spécifique à une étape de l'apprentissage.

- `DataSources/DataSinks` : ouverture et sauvegarde de fichiers de données aux différents formats ;
- `Filters` : sélection et transformation d'attributs ;
- `Classifiers` : algorithmes d'apprentissage supervisé, par catégorie : probabilistes (*Bayes*), par optimisation (*functions*), paresseux (*lazy*), fusion de classifieurs (*meta*), arbres (*trees*), par règles (*rules*) ;
- `Clusterers` : algorithmes d'apprentissage non supervisé ;
- `Evaluation` : outils d'évaluation des algorithmes, qui comprennent en particulier un outil pour sélectionner la classe à predire (*ClassAssigner*) et pour générer des ensembles d'apprentissage et de test.
- `Visualization` : outils pour visualiser les données, les résultats de l'apprentissage et ceux de l'évaluation.

3 Apprentissage d'un SVM

(données et sujet de l'ENSTA)

3.1 Données linéairement séparables

Le format de ficher `arff` est celui natif de Weka. Ouvrez le fichier `SepLineaire.arff` avec un éditeur de texte et observer comment sont codées les informations. Ce fichier contient 40 données, décrites par trois attributs : les coordonnées (x, y) des points et la classe (positif ou négatif). Chargez maintenant le jeu de données `SepLineaire.arff` dans Weka avec le module Arff Loader : un clic gauche pour sélectionner le module, un autre pour le déposer (pas de glisser-coller), clic droit sur le module pour sélectionner 'Configure...'.

1. Visualisez les données grâce à la fonction *Data Visualizer* de Weka (déposer le module *DataVisualizer*, clic gauche sur *ArffLoader*, sélectionner 'DataSet', puis clic droit sur *DataVisualizer*, faites *StartLoading* dans *ArffLoader*). Exhibez plusieurs droites montrant que les points positifs et négatifs sont séparables linéairement.
2. Utilisez les modules *TrainSetMaker* et *TetSetMaker* pour transformer le jeu de donnée en un jeu d'apprentissage et un jeu de test. Rreliez ces modules au module SMO (une implémentation possible des SVM choisie dans Weka, dans les fonctions *Classify*). Dans les paramètres de la méthode SMO, vous constaterez que, par défaut, un noyau polynomial de degré 1 (noyau linéaire) est utilisé, avec une constante C qui vaut 1, et que les données sont normalisées. Les donnés ne nécessitant pas de normalisation, changez l'option pour qu'aucun filtre ne soit appliqué aux données. Fermez la fenêtre de paramètres et lancez la méthode SMO. Observez les résultats. Vous devez y trouver l'équation cartésienne de la droite inférée, les statistiques sur les données d'apprentissage (qui, ici, servent également de données de test). Quel est la valeur du risque empirique ?
3. utiliser les mêmes données pour le test et l'apprentissage n'est pas satisfaisant pour évaluer correctement les performances. Explorer les possibilités offertes par les module *TrainTestSplitMaker* et *CrossValidationFoldMaker*

3.2 Données non linéairement séparables

De même que pour l'exercice précédent, le fichier `SepNonLineaire.arff` contient des données définies dans le plan $X \times Y$. Mais celles-ci ne sont pas séparables linéairement.

1. Cherchez les paramètres de SMO permettant au mieux d'apprendre les données d'apprentissage, également utilisées comme données test dans un premier temps. Comptez à chaque fois le nombre de vecteurs de supports impliqués.

4 Apprentissage d'un arbre de décision

Beaucoup d'autres classifieurs sont disponibles dans Weka. Vous pouvez en particulier tester les arbres de décision vus en cours.

4.1 Construction et évaluation d'arbres

1. Ouvrez le fichier `weather.nominal.arff` à l'aide d'un éditeur de texte. Combien y a-t-il d'instances ? d'attributs ? Quels sont les types des attributs ? Quelle est la classe à prédire ?

2. Mettez en place un processus d'apprentissage pour classer ces données grâce à un arbre de décision de type C4.5 (J48 dans Weka), avec un jeu de données utilisant (outil *trainTestSplitMaker*) 66% des données pour l'apprentissage et le reste pour le test.
3. Visualisez les résultats : combien de données de test ont été bien classées ? Pour quelle classe l'apprentissage est-il le plus/moins efficace ?
4. Faites varier la quantité de données d'apprentissage et la graine [2] utilisée. Commentez les résultats. Comment améliorer la stratégie d'évaluation ?
5. Utilisez maintenant le fichier `weather.arff`. Quelle est la différence entre ces deux jeux de données et comment cela se ressent sur l'algorithme et les résultats obtenus ?

5 Élagage et simplification

Par défaut, l'arbre J48 est élagué. L'élagage est réalisé grâce à un ensemble de validation choisi de manière (presque) transparente pour l'utilisateur.

1. Ajoutez à votre graphe un arbre J48 non élagué (paramètres : *unpruned = true* -non élagué- et *minNumObj=1* -1 exemple minimum par feuille-). Cet arbre n'utilise pas de données de validation.
2. Comparez les structures des arbres obtenus et leurs performances. Quel phénomène observe-t-on ici ?
3. Autorisez plus d'exemples (2 ou 3) par feuille sur l'arbre non élagué (paramètre *minNumObj*). Comment peut-on interpréter la différence de résultat ?

6 Apprentissage bayésien

6.1 Bayes naïf

La méthode d'apprentissage se trouve dans le répertoire `Classifiers :Bayes :NaiveBayes`.

Mettez en oeuvre et testez un apprentissage par méthode de Bayes naïve sur l'ensemble de données `weather.arff`.

1. Rappelez ce qu'est l'hypothèse de Bayes naïve utilisée ici.
2. À quoi correspondent les résultats fournis par le modèle ? Commentez les résultats par rapport à ce qui a été vu en cours.
3. Calculez la classe de la donnée `[Ensoleillé, 73, 81, Fort]` avec les données fournies par Weka [3] et vérifiez le résultat expérimentalement.
4. testez avec le jeu de données `weather.arff`, comparez les résultats avec ceux des arbresw de décision.

---

2. graine (*seed*) : valeur utilisée pour initialiser le tirage aléatoire des données d'apprentissage et de test. À une valeur donnée correspond un ordre de tirage des données.

3. Utilisez un tableur pour accélerer l'application numérique des calculs, notamment par la fonction LOI.NORMALE(x ; μ, σ, FAUX).

6.2 Approche non paramétrique

On s'intéresse à présent aux algorithmes de type ppv, comme `IB1` ou *1-NN* (1 plus proche voisin ou *1 Nearest Neighbour*) qui se trouve dans le répertoire `Classifiers :Lazy`.

1. Expliquez le fonctionnement de l'algorithme `IBk`.
2. Analysez les résultats obtenus en utilisant l'algorithme `IB1` sur l'ensemble de données `iris.arff`.
3. Répétez avec un 2-NN. Qu'observez-vous ? Interprétez le résultat.
4. Comment améliorer le résultat ?

7 Fusion de classifieurs

Nous verrons dans un prochain cours (en fonction du temps qu'il reste) certaines techniques pour fusionner des classifieurs. Weka met à disposition ces techniques, notamment celles inspirées du *Boosting*.

1. Utilisez l'algorithme *adaBoost* pour réaliser la fusion de 50 (paramètre *numIterations* d'adaboost) arbres de décision (classifieur *J48*).
2. Comparer la structure et les performances de classification avec celles de votre arbre de décision initial. Commentez.