TP RF 3 : Weka

Dans ce TP, inspiré de celui d'A. Cornuéjols, nous utiliserons le logiciel libre Weka écrit en java pour réaliser des expériences d'apprentissage utilisant :
• les réseaux de neurones
• les Séparateurs à Vastes Marges (SVM)
• le Boosting
Vous trouverez le site donnant des détails sur Weka (et un manuel en ligne) à l'adresse :
http://www.cs.waikato.ac.nz/ml/weka/

Pour lancer Weka, lancer une machine virtuelle développement : VM_Productions \ VMWARE WINDOWS 7 PRO 64 bits - DEVELOPPEMENT - HUBERT CARDOT.
Les bases de données (.arff) sont disponibles sur Célène et dans le répertoire C:\Program Files\weka-3-8\data\.

1. Prise en main sur la base iris.arff
Dans un premier temps, vous allez vous familiariser avec l'utilisation de Weka en utilisant la base de données iris.arff. Il s'agit d'une base de données, très célèbre, comportant 150 exemples de fleurs décrites par 5 attributs à valeur continue et appartenant à 3 classes.
1. Recopier le fichier de données iris.arff dans un répertoire de votre choix.
2. Lancez Weka. Dans la fenêtre « Weka GUI chooser » qui s'est ouverte, cliquez sur le bouton « Explorer ». Une fenêtre « Weka Explorer » doit s'ouvrir. C'est elle qui nous servira d'interface principale.
3. Cliquez sur le bouton « Open file… ». Choisissez le fichier de données, ici, « iris.arff ».
Un certain nombre d'informations vont apparaître dans la fenêtre « Weka Explorer » : le nombre d'exemples, le nombre d'attributs, …. En bas à droite, la répartition des trois classes en fonction de la première variable de description : « sepallength ». Si, dans la fenêtre en bas à gauche, vous cliquez sur un autre attribut, vous verrez apparaître en bas à droite la répartition de la classe en fonction de cet attribut.
4. Nous allons maintenant tester une méthode d'apprentissage sur ces données. Pour cela, allez sur l'onglet « Classify ». Puis choisissez une méthode d'apprentissage grâce au bouton « Choose ». Allez dans le répertoire Classifiers : Functions : MultiLayerPerceptrons afin de tester la méthode du perceptron multicouche.

Les **principales options disponibles dans l'Explorer** (pour les Perceptron multicouches) sont les suivantes (cliquer dans le champ à droite du bouton Choose) :

• *GUI* : permet l'utilisation d'une interface graphique (pour plus de détails, lire la doc en ligne (bouton more dans la fenêtre de choix des options)).
• *autobuild* : Connecte les couches cachées : le laisser à True.
• *decay* : si vrai, faire décroître le taux d'apprentissage : les poids sont moins modifiés au fur et à mesure de l'apprentissage.
• *hiddenLayers* : permet de décrire le nombre et la taille des couches cachées. La description est :
– Soit une suite d'entiers (le nombre de neurones par couche) séparés par des virgules.
– Soit les valeurs spéciales déterminant une seule couche cachée :
  – a : (nombre d'attributs+nombre de classes)/2
  – i : nombre d'attributs
  – o : nombre de classes
  – t : nombre d'attributs+nombre de classes
• *learning rate* : le η du cours info bio-inspirée
• *momentum* : le α du cours info bio-inspirée : $\Delta w(n) = \eta\, wx + \alpha\, \Delta(n-1)$. Pour observer l'algorithme de retro-propagation du gradient dans toute sa pureté, fixez-le à 0.

• *nominalToBinaryFilter* : transforme les attributs nominaux en attributs binaires : un attribut pouvant prendre k valeurs différentes sera transformé en k attributs binaires, un seul de ces attributs valant 1. Je ne sais pas comment fait Weka lorsque cette option est mise à False !
• *normalizeAttributes* : les valeurs des attributs (y compris les attributs nominaux qui seront passés dans le filtre nominalToBinaryFilter) seront toutes ramenées entre -1 et 1.
• *normalizeNumericClass* : si la classe est numérique, on la normalise (de façon interne) : permet d'améliorer les résultats.
• *training time* : le nombre de fois (epochs en anglais) où l'on fera passer l'ensemble d'apprentissage à travers le réseau.

5. Réalisez un apprentissage à l'aide d'un perceptron multicouche sur la base iris.arff. Pour cela, après avoir choisi la méthode MultilayerPerceptron, cliquez sur la cellule à droite de « Choose », c'est-à-dire sur le nom de la méthode sélectionnée, ici MultilayerPerceptron. Les options possibles sont alors affichées, vous offrant des choix correspondant aux options ci-dessus.
Après apprentissage, qui peut prendre un peu de temps, vous devriez obtenir le résultat dans la fenêtre de droite « Classifier output ».
6. Analyse du résultat. Si vous avez utilisé la méthode de validation croisée, le haut de la fenêtre vous donne le détail du réseau de neurones appris sur le premier jeu de données. Vous devriez obtenir ensuite un tableau de résultats du type :

```
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances         146               97.3333 %
Incorrectly Classified Instances         4                2.6667 %
Kappa statistic                          0.96
Mean absolute error                      0.0327
Root mean squared error                  0.1291
Relative absolute error                  7.3555 %
Root relative squared error             27.3796 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0          1         1         1          1        Iris-setosa
                 0.96      0.02       0.96      0.96      0.96       0.996    Iris-versicolor
                 0.96      0.02       0.96      0.96      0.96       0.996    Iris-virginica
Weight. Av 0.973     0.013      0.973     0.973     0.973      0.998

=== Confusion Matrix ===
  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 48  2 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

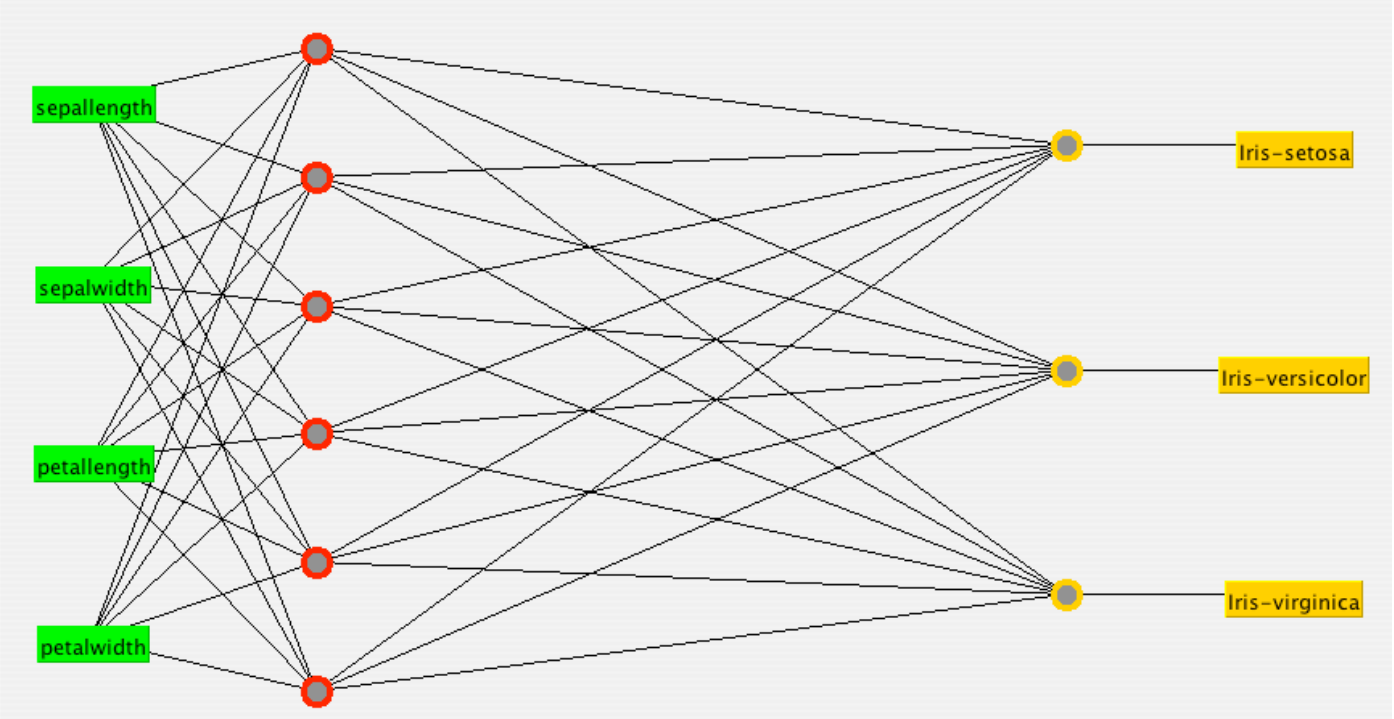
D'après ce tableau, on voit que 97,3 % des exemples ont été classés correctement.
1.1 Effets du choix des valeurs de paramètres sur les résultats
Question 1 : En demandant de visualiser les erreurs (click droit dans Result List, puis Visualizeclassifiers errors), retrouvez les instances sur lesquelles le classificateur se trompe.
Question 2 : Relancer l'apprentissage, en fixant maintenant un temps de calcul (training time) de 25 epochs, au lieu de 500. Regardez de nouveau où se situent les instances mal classées.
Question 3 : Faire des essais en modifiant aussi le learning rate.

1.2 Analyse des réseaux de neurones
Reprenez un apprentissage de 500 epochs, mais demandez maintenant la visualisation du réseau (GUI dans la fenêtre des options du **MultilayerPerceptron**). Maintenant, lorsqu'on appuie sur Start une fenêtre s'ouvre, qui nous montre le réseau construit par **Weka.**
La couche cachée contient 3 neurones (choix par défaut de Weka ((4 + 3)/2 = 3)
Pour lancer l'apprentissage, il faut maintenant, dans la fenêtre représentant le réseau, appuyer sur « Start », puis sur « Accept » pour fermer la fenêtre et obtenir les statistiques dans la fenêtre Classifier output.

Les nœuds sont numérotés de 0 à *n*, en commençant par la couche la plus à droite, et par le neurone du haut. Après apprentissage, la fenêtre Classifier « output » nous donne la valeur des poids dans tout le réseau. Par exemple, les poids en entrée du nœud « node 0 », celui qui correspond à la sortie « iris setosa », ont les valeurs indiquées ci-après :

```
Sigmoid Node 0
Inputs Weights
Threshold -2.428722952755603
Node 3 -0.4668883326080196
Node 4 6.159683060398803
Node 5 -3.9410809046292963
```

Question 4 : A partir de ces informations en chaque nœud, pourrait-on lire ce que le réseau a appris, ou au moins, pouvoir dire ce que chaque neurone calcule ?
Question 5 : En étudiant le rôle de chaque neurone interne, pensez-vous que l'on ait vraiment besoin de trois neurones dans la couche interne?
Question 6 : Nous allons maintenant faire varier le nombre de neurones de la couche cachée. Pour cela, cliquez sur le bouton « MultilayerPerceptron … ». Puis mettez « autobuild » à true. Puis choisissez le nombre de neurones de la couche cachée dans hidden layer. Par exemple, si vous mettez 6, et que vous mettez « GUI » à True, vous obtiendrez le réseau suivant :



Vous devriez observer que le nombre de neurones en couche cachée n'affecte pas beaucoup les performances en généralisation. Cela devrait être différent avec d'autres jeux de données.

2. Mêmes expériences avec d'autres jeux de données
On refera les mêmes expériences avec d'autres jeux de données dont :
- tic-tac-toe.arff
- credit.arff

3. Comparaison avec un autre algorithme d'apprentissage : les SVM
Utilisez maintenant l'algorithme SVM pour les mêmes expériences.
Pour cela, il faut choisir : classifiers : functions : SMO
Question 7 : Est-ce que les erreurs de classification commises sont les mêmes que pour le perceptron multicouche ?
Question 8 : Modifier les paramètres pour prendre un kernel gaussien (RBF) et ajuster C et sigma (1/gamma). Trouver le jeu de paramètres pour faire 100 % sur la base Tic-tac-toe.

4. Utilisation du boosting

Nous allons maintenant utiliser la méta-méthode d'apprentissage du boosting au-dessus d'arbres de décision.

Pour cela, il faut choisir : classifiers : meta : AdaBoostM1

Question 9 : Refaites des expériences en modifiant les valeurs des paramètres. Obtenez-vous une amélioration des performances ?

Question 10 : Recommencer avec ADTree comme classificateur de base.