Decision Trees

Haute école de gestion de Genève
Informatique de Gestion
Alexandros Kalousis

Decision Trees - Rpart

- We will use the package *rpart*
- You should make it available in your R environment executing the command: *library(rpart)*
- We will use the decision tree package in two modes:
  - *Training:* Here we have a training dataset and we apply the decision tree algorithm to produce a predictive model.
  - *Application:* Here we apply the learned decision tree on one or more instance in order to predict the class label.
- The application mode typically appears either
  - in *testing* mode, i.e. when we evaluate and test our decision tree, to see whether it has a good predictive performance.
  - or in *production* mode, i.e. when the model is deployed in the business.

Training a decision tree (1)

- We have a training dataset
- We apply the rpart decision tree algorithm on it
- and we get a decision tree as a result

```
#read data
myData<- read.table("data/iris.csv",header=T,sep=",")

#create the training dataset
#get 50% of the data for training
trainIndex <- sample(1:dim(myData)[1],size=0.5*dim(myData)[1])
trainData  <- myData[trainIndex,]


#Train a Decision Tree on myData.
#The parameter:
#  formula=type.~.
#sets the target/class variable to be the type.
#and use as predictive variables all the others
dt<-rpart(formula=type.~.,data=trainData)
```

Training a decision tree (2)

- The syntax of the training command is common also to other classification algotithms
- We can use the *formula* parameter to specify which attributes to use as predictive attributes

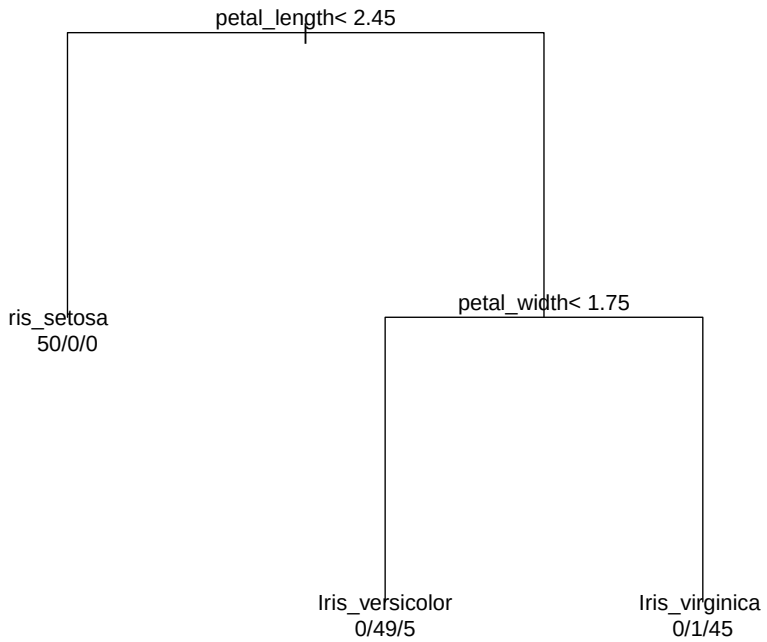
```
#The parameter
# formula=type.~sepal_length+petal_width
#sets the target/class variable to be the type.
#and uses as predictive:
# the sepal_length
# and petal_width
dt<-rpart(formula=type.~sepal_length+petal_width,
          data=trainData)
```

Visualizing a decision tree (1)

```
#To see the decision tree in
#text form just type it:
> dt
n= 150

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 150 100 Iris_setosa (0.33 0.33 0.33)
  2) petal_length< 2.45 50   0 Iris_setosa (1.00 0.00 0.00) *
  3) petal_length>=2.45 100  50 Iris_versicolor (0.00 0.50 0.50)
    6) petal_width< 1.75 54   5 Iris_versicolor (0.00 0.90 0.09) *
    7) petal_width>=1.75 46   1 Iris_virginica (0.00 0.021 0.97) *

#after the test of each node you have
#  the number of instances in the node
#  the number of instance wrongly classified
#  the class label assigned to the node
#  in parenthesses (   ) the percentages of the different classes
```

Visualizing a decision tree (2)

```
#Or use the plot command to produce a
#graphical output
plot(dt,compress=T,uniform=T,margin=0.2)
text(dt,digits=3,use.n=T)

#in a leaf the triplet x/y/z/... shows the
#number of instance of the first/second/third/... class
```



Rpart, other important parameters

- *parms=list(split='information')*: use information gain as selection criterion.
- *minsplit*: the minimum number of observations that must exist in a node in order for a split to be attempted.
- *minbucket*: the minimum number of observations in any terminal <leaf> node. If only one of minbucket or minsplit is specified, the code either sets minsplit to minbucket*3 or minbucket to minsplit/3, as appropriate.
- *cp*: complexity parameter. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted
- IMPORTANT: the three last parameters control the size/complexity of the tree

Rpart, Testing

```
#Creating the testing data
#(actually the rest of the data)
testData <- myData[-trainIndex,]

#to make it even more reallistic
#hide the labels
testDataNoLab <- testData[,1:4]

#get the predictions
predictions <- predict(dt,testDataNoLab,type="class")
#             1               4               5               9               1
#    Iris_setosa     Iris_setosa     Iris_setosa     Iris_setosa     Iris_setos
#            19              21              22              23               2
#    Iris_setosa

#alternatively, output for
#each testing instance the
#probabiity of each class.
predict(dt,testDataNoLab)
#    Iris_setosa Iris_versicolor Iris_virginica
#1             1      0.00000000     0.00000000
#4             1      0.00000000     0.00000000
#5             1      0.00000000     0.00000000
#
```

Testing, accuracy computation

```
#Compare the true labels of the
#testing instances with the predictions
CorrectWrong <- (predictions==testData[,5])

#Get the Correct (i.e. TRUE above)
which(CorrectWrong)

#Get the Number of Correct
numCorrect<-length(which(CorrectWrong))

#Get the accuracy, i.e. the % of correct
numCorrect/dim(testData)[1]
[1] 0.9466667
```

Retrieve decision tree size (number of leaves)

```
# To find the number of leaves in an rpart decision tree
# in R, you can use the dt$frame component of your rpart
# object.  This line of code counts the number of times
# the string "<leaf>" appears in the var column of the
# dt$frame data frame,  which corresponds to the number
# of leaves in the decision tree.
numLeaves <- sum(dt$frame$var == "<leaf>")
leaves[rowIndex,colIndex] <- numLeaves
```

Retrieve decision tree size (number of nodes)

```
# To determine the number of nodes in an rpart decision
# tree in R, you can use the max() function on the
# where component of your rpart object. The where component
# contains the terminal node number for each observation,
# so by finding the maximum value, you get the total number
# of nodes. Here's an example:
numNodes<- max(dt$where)
nodes[rowIndex,colIndex] <- numNodes
```

Retrieve decision tree size (depth)

```
# To find the depth of an rpart decision tree in R,
# you can use the  unexported function tree.depth
# from the rpart package.  This function calculates
# the depth of each node in the tree. Here's how
# you can use it to find the maximum depth of
# the tree:
# Assuming 'dt' is your rpart model
theNodes <- as.numeric(rownames(dt$frame))
depth <- max(rpart:::tree.depth(theNodes))
depths[rowIndex,colIndex] <- depth
```

Exercice sur les arbres de décision I

- Dans ce TP, votre objectif est de :
  - entraîner et évaluer l'algorithme de l'arbre de décision sur l'ensemble de données du projet
  - explorer comment les différents hyperparamètres affectent les performances prédictives
  - explore comment les différents hyperparamètres affectent la taille de l'arbre
  - et entraîner et expliquer les modèles finaux

Exercice sur les arbres de décision II

- Comme toujours : script générique, appelez runAnalysis(...) pour faire tout le travail.
- Vous devriez expérimenter avec l'index *Information gain* et *Gini* comme critère de sélection d'attribut.
- Livrables :
  - Le code R exécutable et générique (script .R) doit s'exécuter sans erreur.
  - Rapport PDF dans lequel vous présenterez vos résultats (il ne devrait y avoir aucun code dans le rapport..

Exercice sur les arbres de décision III

- Pour effectuer l' apprentissage et le test, vous devez :
  - divise l'ensemble de données d'entraînement d'origine en deux parties, les deux tiers que vous utiliserez pour l'entraînement et le tiers restant que vous utiliserez pour les tests.
  - Vous devez répéter la procédure cinq fois avec différentes répartitions aléatoires et rapportez l' *accuracy (TBC) moyenne*.
  - Assurez-vous de rendre la procédure d'évaluation *générale* car vous en aurez besoin pour évaluer d'autres algorithmes d'apprentissage.
  - Assurez-vous d'avoir des répartitions aléatoires *reproductibles*.

Exercice sur les arbres de décision IV

- Vous devez évaluer les performances avec différents réglages de paramètres, à savoir :

$$minsplit \times cp \in (\{5, 10, 50, 100, 200\} \times \{0,001, 0,01, 0,1, 0,2\})$$

et rapportez la précision moyenne pour chaque combinaison de paramètres à la fois pour le gain d'information et l'indice de Gini, en utilisant un tableau de la forme :

	0.001	0.01	0.1	0.2
5				
10				
50				
100				
200				

expliquez quelle combinaison d'hyperparamètres vous allez utiliser pour entraîner le modèle final et pourquoi.

Exercice sur les arbres de décision V

- Étudier l'effet des paramètres de complexité du modèle (pour un gain d'information uniquement). Pour faire cela:
  - Entraînez le modele en utilisant toutes les données disponibles.
  - Construize trois matrices comme suit :

	0,001	0,01	0,1	0,2
5				
10				
50				
100				
200				

  dans lequel pour chaque combinaison de $min - split$ et cp vous donnerez le nombre de feuilles de l'arbre, le nombre de nœuds et sa profondeur.
  - Montrez graphiquement comment la taille de l'arbre (# nœuds, # feuilles, profondeur) change en fonction du minsplit pour un cp fixe et vice versa.
  - Expliquez vos observations et comment les valeurs de ces paramètres affectent la taille de l'arbre.

Exercice sur les arbres de décision VI

- Les modèles finaux :
  - Vous devez construire deux modèles finaux, un avec gain d'information et un avec l'indice gini, sur l'ensemble de données en utilisant les combinaisons de paramètres qui avaient le plus haut précision pour chacun des deux critères de sélection des attributs. Expliquez comment le deux arbres de décision classent les instances, quels sont les attributs importants et comment ils affectent la classification ?
  - Les deux arbres finaux sont-ils identiques ? Sinon, utilisent-ils le même ensemble d'attributs ? S'il y a des différences, pourquoi ?
  - Avec votre meilleur modèle, fournissez une prédiction pour une instance que vous avez sélectionnée au hasard dans l'ensemble de données.