Naive Bayes

Haute école de gestion de Genève
Informatique de Gestion
Alexandros Kalousis

Naive Bayes

- We will use the package *e1071*
- You should make it available in your R environment executing the command: *library(e1071)*
- As with the decision tree algorithm we will also use the Naive Bayes 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.

Installing the Naive Bayes package (e1071)

- If the package e1071 is not available you should install it using

```
setInternet2(TRUE)
install.packages("e1071",lib="p:\\Mes Documents",
                         destdir="p:\\Mes Documents",
                         repos="http://lib.stat.cmu.edu/R/CRAN")

library("e1071",lib.loc="p:\\Mes Documents")

#if you are working in your machine the name of the package is enough
#otherwise you need to give the directories to avoid permission problems
```

Training a Naive Bayes Model, continuous dataset

- We have a training dataset
- We apply the naive Bayes algorithm algorithm on it
- and we get a naive Bayes model 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 Naive Bayes on myData.
#The parameter:
#  formula=type.~.
#sets the target/class variable to be the type.
#and use as predictive variables all the others
nb<-naiveBayes(formula=type~.,data=trainData)
```

Visualizing a Naive Bayes Model on Continuous Data

```
#To see the naive bayes in
#text form just type it:
> nb
A-priori probabilities:
Y
    Iris_setosa Iris_versicolor  Iris_virginica   => P(type.) = {P(Iris_setosa) P(Iris_versicolor)  P(Iris
      0.3466667       0.3600000       0.2933333

Conditional probabilities:
                 sepal_length
Y                     [,1]      [,2]  => column[,1] is the conditional mean of sepal_length for each class
  Iris_setosa     5.065385 0.3224187     column[,2] is the standard deviation of sepal_length for each cla
  Iris_versicolor 5.844444 0.4676729
  Iris_virginica  6.663636 0.6366171

                 sepal_width
Y                     [,1]      [,2] => column[,1] is the conditional mean of sepal_width for each class
  Iris_setosa     3.542308 0.3360632     column[,2] is the standard deviation of sepal_width for each clas
  Iris_versicolor 2.722222 0.2606697
  Iris_virginica  2.959091 0.3620827

                 petal_length
Y                     [,1]      [,2] => ...
  Iris_setosa     1.469231 0.1975231
  Iris_versicolor 4.203704 0.4879167
  Iris_virginica  5.586364 0.6057624

                 petal_width
Y                     [,1]      [,2] => ...
  Iris_setosa     0.2807692 0.1166850
  Iris_versicolor 1.3037037 0.1764642
  Iris_virginica  1.9863636 0.3028365
```

Training a Naive Bayes Model, discrete dataset

- As before

```
#read data
myData<- read.table("data/titanic.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 Naive Bayes on myData.
#The parameter:
#  formula=result~.
#sets the target/class variable to be the type.
#and use as predictive variables all the others
nb<-naiveBayes(formula=result~.,data=trainData)
```

Visualizing a Naive Bayes Model on qualitative Data

```
#To see the naive bayes in
#text form just type it:
> nb
A-priori probabilities:
Y
     mort      surv      => P(result)= {P(mort) P(Surv)}
0.6636364 0.3363636

Conditional probabilities:
      class                                          => P(class | result)
Y            crew      first     second      third
  mort 0.44383562 0.08082192 0.10958904 0.36575342  => P(crew|mort) P(first|mort) ...
  surv 0.30000000 0.29459459 0.14864865 0.25675676  => P(crew|surv) P(first|surv) ...

      age
Y             adu        enf
  mort 0.96027397 0.03972603
  surv 0.91351351 0.08648649

      sex
Y               f          m
  mort 0.07671233 0.92328767
  surv 0.47567568 0.52432432
```

Accessing a Naive Bayes Model

```
#retrieve the counts of the target class
nb$apriori

#output:
# Y
# mort surv
# 741  359

#retrieve the conditional distributions (condition propabilities for disrete attributes,
#or class conditional means and standard deviations for continuous) of attributes given class
nb$tables

#output
#$class
#      class
#Y            crew      first     second      third
#  mort 0.42105263 0.08636977 0.11605938 0.37651822
#  surv 0.27576602 0.29247911 0.17827298 0.25348189
#
#$age
#      age
#Y             adu        enf
#  mort 0.95951417 0.04048583
#  surv 0.90529248 0.09470752
#
#$sex
#      sex
#Y               f          m
#  mort 0.07152497 0.92847503
#  surv 0.48467967 0.51532033
```

Accessing a Naive Bayes Model, retrieve the conditional distribution for a particular attribute

```
#We can access the elements of nb\$tables list either by their name,
#e.g. the class conditional probabilities for the age variable
nb$tables$age

#      age
#Y             adu        enf
#  mort 0.95951417 0.04048583
#  surv 0.90529248 0.09470752

#or by their index
#e.g. same as above but with the index instead of the name
nb$tables[[2]]
#      age
#Y             adu        enf
#  mort 0.95951417 0.04048583
#  surv 0.90529248 0.09470752
```

NaiveBayes, Testing and Accuracy Estimation

- As with decision trees we predict on a test set using the predict function, i.e.:

```
#create the test dataset
#get the remaining 50% of the data for testing
testData  <- myData[-trainIndex,]

predict(nb, testSet)
#outputs:
#[1] mort mort mort mort mort mort surv mort surv mort mort surv mort mort
#[15] mort mort surv mort mort mort mort surv surv mort surv surv mort mort

# or to have also the class probabilities for every instance

predict(nb, testSet, type="raw")

#outputs:
#            mort      surv
# [1,] 0.85751563 0.1424844
# [2,] 0.85411860 0.1458814
# [3,] 0.85751563 0.1424844
# [4,] 0.85751563 0.1424844
# [5,] 0.53788886 0.4621111
# [6,] 0.53788886 0.4621111
# [7,] 0.08703797 0.9129620
# ...
```

- For testing and accuracy estimation procedures refer to the corresponding procedures in the decision tree exercise.

Naive Bayes Exercise I

- Dans cet exercice, vous devriez:
  - entraîner l'algorithme de naive Bayes sur l'ensemble de données,
  - tester la performance (taux de bien classes) de naive Bayes
  - et comprendre/expliquer le modèle final appris sur la totalite des données et la façon dont il est utilisé pour classer les instances.
- Vous devez également créer un *classificateur par défaut*, c'est-à-dire le classificateur qui prédit toujours comme classe la classe majoritaire dans l'ensemble *train*. Comparez les performances de Naive Bayes à celles du classificateur par défaut et commenter.
- Attention : lors de la lecture des données avec *read.table*, utiliser le paramètre *colClasses* pour définir correctement les types de variables.

Apprentissage et estimation de performance/testing

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' *TBC moyenne*.
- Assurez-vous de rendre la procédure d'évaluation *générale* car normallement vous en aurez besoin pour évaluer d'autres algorithmes d'apprentissage.
- Assurez-vous d'avoir des répartitions aléatoires *reproductibles*.

Le classificateur par défaut

- Le classificateur par défaut est un classificateur très simple qui prédit toujours la classe majoritaire pour n'importe quelle instance.
- Il nous permet d'évaluer si d'autres modèles apries sont vraiment utile, en comparant leurs performances à celles du classificateur par défaut
- Vous devez créer deux fonctions :
  - trainDefaultClassifier : prend en entrée un ensemble d'entraînement et l'index de la variable cible et renvoie la classe majoritaire
  - predictDefaultClassifier : prend en entrée un ensemble de test et renvoie le vecteur respectif avec les prédictions.

Comprendre et expliquer le modèle de naive Bayes I

- Comment naive Bayes gèrent-il les attributs prédictifs qualitatifs?
- Utiliser les barplots pour visualiser:
  - la distribution conditionnelle de la classe, c'est-à-dire $P(A|C)$, d'un attribut qualitatif A de votre choix
  - la distribution a priori de l'attribut de classe, c'est-à-dire $P(C)$,

  qui sont apprises par l'algorithme de Naive Bayes. Comment ces distributions sont-elles liées à celles que nous avons estimées dans le premier TP?

Comprendre et expliquer le modèle de naive Bayes II

- Que se passe-t-il avec les attributs quantitatifs, comment Naive Bayes modélise-t-il le $P(A|C)$ dans ce cas ?
- Quelles quantités calcule-t-il pour les attributs quantitatifs et comment ces quantités sont-elles liées à celles que nous avons calculées dans le premier exercice ?
- Utilisez un attribut *quantitatif* de votre choix pour expliquer ce que Naive Bayes fait avec les attributs quantitatifs.

Classification des instances avec le modèle naive Bayes

- Expliquez comment les densités conditionnelles que vous avez décrites dans les deux points précédents (attributs qualitatifs et quantitatifs) sont utilisées par le modèle de Naive Bayes pour effectuer la classification d'une instance.
- Choisissez au hasard une instance d'apprentissage et expliquez comment le modèle appris par Naive Bayes va la classer en montrant ce qui se passe à l'aide des deux attributs que vous avez choisis ci-dessus et comment les différentes parties du modèle sont utilisées pour effectuer cette classification.