Introduction to Artificial Intelligence (ARIN)

Naive Bayes with R

This practical introduces you to the naive Bayes model using the R statistical environment. It does not assume prior knowledge of R. I know everything works in Linux so switch if you are not running Linux already.

1. Starting up R
2. Exploring the Iris dataset
3. Constructing a naive Bayes classifier
4. Understanding naive Bayes

Starting up R

To invoke R, just type "R" at the Linux command line. To get R documentation in a web browser type:

```
help.start()
```

at the R prompt. There's no need to study all this documentation right now. There are fancier ways of working with R (I run it from with in an emacs buffer) but just typing commands at the R prompt is nice and simple, so that's what we'll use here. You can scroll through recent commands to R using the up and down arrow keys on your keyboard. When you're done with an R session enter the command:

```
q()
```

You are then given the option of saving any objects you have created (your 'workspace'). If you say yes, then the workspace is saved in a file called .RData in your working directory. Your commands are also saved (without asking you!) in a file called .Rhistory, ready for any future R session.

Exploring the Iris dataset

In this practical you will explore the classic "Iris" dataset. For this we will draw on the excellent Data Mining Algorithms in R Wikibook, which includes a useful section on naive Bayes.

The Iris dataset is pre-installed in R, since it is in the standard datasets package. To access its documentation, click on 'Packages' at the top-level of the R documentation, then on 'datasets' and then on 'iris'. As explained there there are 150 datapoints and 5 variables. Each datapoint concerns a particular iris flower and gives 4 measurements of the flower: Sepal.Length, Sepal.Width, Petal.Length and Petal.Width together with the flower's Species. The goal is to build a classifier that predicts species from the 4 measurements, so species is the class variable.

To get the Iris dataset into your R session, do:

```
data(iris)
```

at the R prompt. As always, it makes sense to look at the data. The following R command (from the Wikibook) does a nice job of this.

```
pairs(iris[,1:4],main="Iris Data (red=setosa,green=versicolor,blue=virginica)", pch=21, bg=c("red","green3","blue")[unclass(iris$Species)])
```

The 'pairs' command creates a scatterplot. Each dot is a datapoint and its position is determined by the values that datapoint has for a pair of variables. The class determines the colour of the datapoint. From the plot note that setosa irises have smaller petals than the other two species. Typing:

```
summary(iris)
```

provides a summary of the data and typing:

```
iris
```

prints out the entire dataset to the screen.

Constructing a naive Bayes classifier

We will use the e1071 R package to build a naive Bayes classifier. Firstly you need to download the package (since it is not pre-installed here). Do:

```
install.packages("e1071")
```

Choose a mirror in UK from the menu that will appear. You will be prompted to create a personal R library (say yes) since you don't have permission to put e1071 in the standard directory for R packages.

To (1) load e1071 into your workspace (2) build a naive Bayes classifier and (3) make some predictions on the training data, do:

```
library(e1071)
classifier<-naiveBayes(iris[,1:4], iris[,5])
table(predict(classifier, iris[-5]), iris[5], dnn =list('predicted','actual'))
```

As you should see the classifier does a pretty good job of classifying. Why is this not surprising?

To see what's going on 'behind-the-scenes', first do:

```
classifier$apriori
```
This gives the class distribution in the data: the prior distribution of the classes. ('A priori' is Latin for 'from before').

Since the predictor variables here are all continuous, the naive Bayes classifier generates three Gaussian (Normal) distributions for each predictor variable: one for each value of the class variable Species. If you type:

```r
classifier$Stables$Petal.Length
```

you will see the mean (first column) and standard deviation (second column) for the 3 class-dependent Gaussian distributions:

```
Petal.Length
iris[, 5] [,1]       [,2]
setosa    1.462 0.1736640
testsetosa 4.260 0.4699110
virginica 5.552 0.5518947
```

You can plot these 3 distributions against each other with the following three R commands:

```r
plot(function(x) dnorm(x, 1.462, 0.1736640), 0, 8, col="red", main="Petal length distribution for the 3 different species")
curve(dnorm(x, 4.260, 0.4699110, add=TRUE, col="blue")
curve(dnorm(x, 5.552, 0.5518947), add=TRUE, col = "green")
```

Note that setosa irises (the red curve) tend to have smaller petals (mean value = 1.462) and there is less variation in petal length (standard deviation is only 0.1736640).

---

**Understanding naive Bayes**

In the previous question you were given a recipe which allowed you to construct a naive Bayes classifier. This was for a case where we had continuous predictor variables. In this question you have to work out what the parameters of a naive Bayes model should be for some discrete data.

The dataset in question is called HairEyeColor and has three variables: Sex, Eye and Hair, giving values for these 3 variables for each of 592 students from the University of Delaware. First have a look at the numbers:

```
HairEyeColor
```

You can also plot it as a 'mosaic' plot which uses rectangles to represent the numbers in the data:

```
mosaicplot(HairEyeColor)
```

Your job here is to compute the parameters for a naive Bayes classifier which attempts to predict Sex from the other two variables. The parameters should be estimated using maximum likelihood. To save you the tedium of manual counting, here's how to use margin.table to get the counts you need:

```r
> margin.table(HairEyeColor, 3)
Sex
  Male Female
279   313
> margin.table(HairEyeColor[,1,3])
Sex
Hair  Male Female
Black 56   52
Brown 143 143
Red    34   37
Blond  46   81
```

Note that Sex is variable 3, and Hair is variable 1. Once you think you have the correct parameters speak to me or one of the demonstrators to see if you have it right. (Or if you can manage it, construct the naive Bayes model using the naiveBayes function and yank out the parameters from the model. Read the documentation to do this!)

---

**Answers**

Last modified: Mon May 13 17:17:44 BST 2013