MS1b: SDM - Practical Sheet 3

Logistic and nonparametric methods

In this practical, we will examine several R functions that allow us to implement logistic and non-parametric classification methods. Load the workspace `PracticalObjects` using something like

```
load(url("http://www.stats.ox.ac.uk/%7Eteh/teaching/datamining/PracticalObjects.RData"))
```

This file contains functions and datasets that we will need use in these classes. Alternatively, download from the website and open locally with `load`. Where packages cannot be loaded, type `install.packages()` and follow the instructions to install a package.

Logistic Regression

In this section we first look at the South African heart disease dataset, `SA.heart`, which is located in the workspace `PracticalObjects`. More information about the dataset is available at
`http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/SAheart.info`.
The data has been taken from a larger dataset, described in Rousseauw et al, 1983, South African Medical Journal, and include 160 samples (`chd=1`) and 302 controls (`chd=0`). The explanatory variables that will be used to fit a logistic regression model include `sbp` (systolic blood pressure), `tobacco` (cumulative tobacco in kg), `ldl` (low density lipoprotein cholesterol), `famhist` (family history of heart disease, present or absent), `obesity`, `alcohol` (current alcohol consumption), and `age` (age at onset). The response variable is `chd` (coronary heart disease). We select these variables from the original dataset:

```
SA.hrt <- SA.heart[,c(2:4,6,8:11)]
```

1. We use the function `glm()` to fit the logistic regression model. First we specify the data structure to the function. The response variable is specified on the left hand side of the $\sim$, and explanatory variables included in the model are specified on the right hand side. The . indicates that all variables in the dataset, specified by the argument `data`, except the response should be used as explanatory variables. The argument `family="binomial"` is used to indicate that the logit link function should be used. `SA.glm <- glm( chd . , data=SA.hrt, family="binomial")`

   The function `summary()` can be used to produce a summary of the results of the model fitting.

   ```
   summary(SA.glm)
   ```

   The Z-ratio of the coefficients are computed using the formula in Section 6.2.5 of the lecture notes. The coefficients of `Intercept`, `tobacco`, `ldl`, `famhist`, and `age` are significantly different from zero at a 5% level.

2. It appears that a subset of the variables might be sufficient to explain the variation in the response variable. The function `step()` is used to find such a subset: it drops the least significant coefficient and refits the model. The coefficients are dropped repeatedly, until no further terms can be dropped.

   ```
   SA.step <- step(SA.glm)
   ```

   ```
   summary(SA.step, direction="backward")
   ```

   The simplified model includes only the variables `tobacco`, `ldl`,`famhist`, and `age` as explanatory variables. Note from the summary results give estimates of the coefficients corresponding to these explanatory variables. The coefficient for tobacco is 0.081. Tobacco is measured in kilograms of total lifetime usage. How will an increase in 1kg life-time tobacco usage influence the odds of coronary heart disease? The odds of coronary heart disease is estimated to increase by exp(0.0807)=1.0840 as the life-time tobacco usage increase by 1kg. If `ldl` increases by one unit, with other variables kept constant, the odds of coronary heart disease increases by exp(0.167584)= 1.182445.

3. We can explore the relationship between `tobacco`, `ldl` and the presence of coronary heart disease graphically. First, we create a plot of `tobacco` against `ldl`, and then we indicate the points that correspond to patients with and without heart disease separately.

   ```
   plot(SA.hrt$ldl, SA.hrt$tobacco, type="n", xlab="ldl", ylab="tobacco")
   ```

   ```
   T.vec <- SA.hrt$chd==1  ##vector that indicates for which observations chd=1
   ```

```
points(SA.hrt$ldl[T.vec], SA.hrt$tobacco[T.vec], pch=2, col=1)

T.vec <- (SA.hrt$chd==0)  ##vector that indicates for which observations chd=0

points(SA.hrt$ldl[T.vec], SA.hrt$tobacco[T.vec], pch=3, col=2)

legend(11, 28, c("chd=1","chd=0"), pch=c(2,3), col=c(1,2))
```

The function `legend()` is used to add a legend to the plot. The first two arguments specify the position of the legend, the third argument is a vector of text values to appear in the legend. Examining the scatterplot reveals that the heart disease samples are associated with higher levels of both tobacco and ldl.

4. The fitted `glm()` object returns the fitted mean values, `fitted.values`, $\hat{\theta}_1, \ldots, \hat{\theta}_n$.

   ```
   SA.glm$fitted.values
   ```

   A sample in the dataset is classified as "chd=1" (heart disease) if the fitted mean response for that sample is greater than 0.5. If the fitted mean response for a sample is 0.5 or less than 0.5 the sample is classified as "chd=0" (no heart disease). The function `ifelse` is used to create a vector with the predicted classes of the observations (look at `?ifelse`).

   ```
   class.vec <- ifelse(SA.glm$fitted.values>0.5, 1, 0)

   SA.glm$fitted.values[1:10]

   class.vec[1:10]
   ```

   A comparison of the first 10 values in `SA.glm$fitted.values` and `class.vec` will make the transition of the vector `SA.glm$fitted.values` to the vector `class.vec` clearer.

   What percentage of the observations is misclassified?

   ```
   SA.error <- sum(class.vec!=SA.hrt$chd)/nrow(SA.hrt)
   ```

5. We fit a logistic regression model for the Cushing's dataset. The data matrix `cush` can be created again, as in practical 3.

   ```
   library(MASS)

   data(Cushings)

   cush <- Cushings[Cushings$Type!="u",]

   cush[,3] <- factor(cush[,3],levels=c("a","b","c"))

   cush[,1] <- log(cush[,1])

   cush[,2] <- log(cush[,2])
   ```

   Remember that each observation in the dataset `cush` is classified to belong to one of three classes. Therefore, we need to use the function `multin()` in the library `nnet` to fit a logistic regression model.

   ```
   library(nnet)

   cush.multin <- multinom(cush[,3]~cush[,1]+cush[,2], data=cush, maxit=250)

   summary(cush.multin)

   cush.pred <- predict(cush.multin)

   table(cush[,3], cush.pred)
   ```

   Are the estimated coefficients significantly different from zero? What is the misclassification rate?

Non-parametric Classification: Cushing's Data

To visualise the decision boundaries for non-parametric classification methods we use a dataset with two continuous variables and one classification factor. The Cushing's syndrome dataset, `cush`, is used as an example here.

1. *k-Nearest Neighbours*
   The function `knn` in the package `class` performs a k-nearest neighbour classification. First, we create a training set and a test data set for the Cushing's data: we place roughly 30% of the observations in the test set and the remaining observations in the training set. Make sure you understand each of the following steps.

   ```
   library(MASS)

   cush <- Cushings[Cushings$Type!='u',]

   cush[,3] <- factor(cush[,3],levels=c('a','b','c'))

   cush.type <- cush[,3]

   cush[,1] <- log(cush[,1])

   cush[,2] <- log(cush[,2])

   library(class)

   number.obs <- nrow(cush)

   train.vec <- runif(number.obs)>0.3

   test.vec <- !train.vec

   cush.train <- cush[train.vec,]

   cush.test <- cush[test.vec,]
   ```

   We need to specify the training set, test set, the vector containing the true classifications of the training set, and the number of neighbours for the function `knn`.

   ```
   cush.knn <- knn(train=cush.train[,1:2], test=cush.test[,1:2],

       cl=cush.train[,3], k=1)
   ```

   The function `knn` returns a factor vector containing the classifications of the test set. Compare the predicted classes of the test set with the true classes.

   ```
   cush.knn

   cush.test[,3]
   ```

   How many observations are accurately classified?

   To visualise the k-nn decision boundaries (for a two dimensional dataset) we use the function `visualise.knn` and specify the number of neighbours. This function is available in the workspace `PracticalObjects`.

   ```
   visualise.knn(cush,k=1)

   visualise.knn(cush,k=3)
   ```

   Change the number of neighbours, `k`, and see how it influences the decision boundaries.

2. *Learning Vector Quantisation*
   The function `lvqdemo`, in the workspace `PracticalObjects`, illustrates the iterations of the LVQ algorithm. The algorithm is given in the lecture notes.

   We have to load the package `deldir` to use this function. The argument `num.codes` specifies the number of prototypes per class and `decrease` gives the rate at which the learning rate α should be decreased to zero. We specify 2 prototypes per class, and a decrease of 0.99.

   ```
   library(deldir)

   lvqdemo(cush, num.codes=2, decrease=0.99)
   ```

   Note that the points are coloured according to the class that they belong to. The prototypes for each class are indicated by the larger filled points. The data is presented one point at a time, and the arrow show how the prototype closest to the data point is updated. When running the algorithm the user is prompted to specify the number of iterations. Start by specifying 10 iterations, but increase it to 50 or 100 if the algorithm converges slowly.

We can adjust the parameters `num.codes` and `decrease`. Examine the influence of smaller values for `decrease` on the rate of convergence of the algorithm.

```
lvqdemo(cush, num.codes=2, decrease=0.90)
```

Non-parametric Classification: Singh Data

The dataset `singh` contains the expression profiles on 52 prostate tumour samples, and 50 normal samples. The Singh data consists of a training set, `singh.train`, and a test set, `singh.test`, used for fitting and performance assessment, respectively. The training set contains about 70% of the samples in the original dataset.

1. *k-Nearest Neighbours*
   We start with the k-nearest neighbours method again, and compute the test error rate for $k = 1$ and $k = 3$ neighbours. The first argument for the function `knn` is the training set, without the factor that contains the true class labels. The notation `singh.train[,-1]` is used to extract all the variables (columns) of the dataset `singh.train` except the first variable.

   ```
   singh.knn.1 <- knn(singh.train[,-1], singh.test[,-1],
       singh.train[,1], k=1)
   misclass <- sum(singh.test$outcome != singh.knn.1)
   test.error.knn.1 <- misclass/nrow(singh.test)
   test.error.knn.1
   singh.knn.3 <- knn(singh.train[,-1], singh.test[,-1],
       singh.train[,1], k=3)
   misclass <- sum(singh.test$outcome != singh.knn.3)
   test.error.knn.3 <- misclass/nrow(singh.test)
   test.error.knn.3
   ```

   Compare the error rates for different numbers of neighbours.

2. *Learning Vector Quantisation*
   Next we look at LVQ classification. An initial codebook for LVQ methods is constructed by the function `lvqinit`. Its first argument is a matrix or data frame of training examples, and the second is the classifications for the training examples. The argument `size` is used to specify the size of the codebook.

   singh.code <- lvqinit(singh.train[,-1],singh.train[,1],size=10)

   The function `olvq1` is then used to move examples in the initial codebook to represent the training set better. The argument `codebk` is used to specify the initial codebook. This function returns a codebook, represented as a list with components `x` and `cl` giving the examples and classes, respectively.

   ```
   singh.olvq <- olvq1(singh.train[,-1], singh.train[,1],
       codebk=singh.code)
   singh.olvq$cl
   ```

   Let us determine the error rate for the training dataset. We use the function `lvqtest` to assess the LVQ classification of data points. The first argument of the function is the codebook returned by the function `olvq1`, and the second argument `test` is used to specify the matrix of samples for which we want to predict the outcome. The function returns the predicted class for each of these samples.

   ```
   train.class <- lvqtest(singh.olvq, test=singh.train[,-1])
   train.class
   ```

   The training error is calculated by dividing the number of misclassifications by the total number of samples in the training set. The test error rate is calculated similarly.

   ```
   LVQ.train.error <- sum(train.class!=singh.train[,1])/
       nrow(singh.train)
   ```

```
test.class <- lvqtest(singh.olvq, test=singh.test[,-1])

LVQ.test.error <- sum(test.class!=singh.test[,1])/

    nrow(singh.test)
```

Why do we expect the test error rate to be higher than the training error? Is this the case here?

Exercises

Logistic Regression: Birdkeeping
Ramsey and Shafer (2002) describe data from a study that investigate birdkeeping as a risk factor for developing lung cancer. The study is based on a case-control study of patients in 1985 at four hospitals in The Hague, Netherlands. They identified 49 cases of lung cancer among patients who were registered with a general practice, who were age 65 or younger, and who had resided in the city since 1965. They also selected 98 controls from a population of residents having the same general age structure. Data were gathered on the following variables for each subject:

```
FM = Sex (FEMALE,MALE)
AG = Age, in years
SS = Sosioeconomic status (HIGH,Low), based on wages
     of the principal wage earner in the household
YR = Years of smoking prior to diagnosis or examination
CD = Average rate of smoking, in cigarettes per day
BK = Indicator of birdkeeping (caged birds in the home for more than
     6 consecutive months from 5 to 14 years before diagnosis
     or examination)
```

It is known that smoking and age are both associated with lung cancer incidence. The research question of interest is the following: after age, socioeconomic status and smoking have been controlled for, is an additional risk associated with birdkeeping?

We use this example as a fairly detailed illustration of how to approach a logistic regression problem in *R*.

1. The data found on the teaching webpage can be read into an *R* data frame using the function `read.table()`. Suppose that we have saved the data in the Windows directory `c:\temp`, then we can read and display the data as follows:

   ```
   birds <- read.table(file="c:\\temp\\birds.txt",sep='\t',header=TRUE)
   attach(birds)
   birds
   ```

   The above command specifies that the data is in tab-delimted format and that the first line contains "headers" for the columns of the table. Study the different variables in the dataset. What does the `attach` function do?

2. A useful first step in any analysis is to perform a graphical exploration of the raw data. A plot of the binary response variable versus an explanatory variable is not worthwhile, since there are only two distinct values for the response variable. An useful alternative is to examine a scatterplot of one of the explanatory variables versus another, with codes to indicate whether the response is 0 or 1. We can construct a coded scatterplot of years of smoking versus age of subject.

   ```
   # set up plot without plotting the data
   plot(x=AG,y=YR,ylab="Years of Smoking",xlab="Age(Years)",
        main="", cex=1.4,,type="n")
   # identify birdkeepers with cancer
   g1 <- BK=="BIRD"&LC=="LUNGCANCER"
   # plot birdkeepers with cancer with filled triangles
   points(x=birds[g1,5],y=birds[g1,6],cex=1.5,pch=17)
   # identify birdkeepers without cancer
   g2 <- BK=="BIRD"&LC=="NOCANCER"
   # plot birdkeepers without cancer with empty triangles
   points(x=birds[g2,5],y=birds[g2,6],cex=1.5,pch=2)
   # identify non-birdkeepers with cancer
   g3 <- BK=="NOBIRD"&LC=="LUNGCANCER"
   # plot non-birdkeeper with cancer with filled circles
   ```

```
points(x=birds[g3,5],y=birds[g3,6],cex=1.5,pch=16)
# identify non-birdkeeper without cancer
g4 <- BK=="NOBIRD"&LC=="NOCANCER"
# plot non-birdkeeper with cancer with filled circles
points(x=birds[g4,5],y=birds[g4,6],cex=1.5,pch=1)
# add a legend to the plot
legend(list(x=37,y=49),legend=c("birdkeeper,cancer",
  "birdkeeper,no cancer","non-birdkeeper,cancer",
  "non-birdkeeper, no cancer"),pch=c(17,2,16,1))
```

What can we derive from this plot?

3. Let us proceed to fit an initial model. It is usually a good idea to start with a fairly *rich* model. What does this mean?

```
options(contrasts=c("contr.treatment","contr.poly"))
birds.glm <- glm(LC~SEX+SS+AG+YR+CD+BK+I(YR^2)+I(CD^2)+YR*CD,
    family=binomial,data=birds)
```

Can you write down the model that we are fitting above?

4. We can extract a summary of the above model as follows:

```
summary(birds.glm,cor=FALSE)
```

What do you learn from the summary output?

5. We can perform an automatic stepwise model selection based on AIC as follows:

```
library(MASS)
# perform stepwise model selection based on AIC
birds.step <- stepAIC(birds.glm,trace=FALSE)
# display the results of stepwise selection
birds.step$anova
# display details of the final model
summary(birds.step,cor=F)$coef
```

6. Does the final model make sense intuitively?

7. How do we interpret this model?

8. We can and should perform some model checking for this model. Below we plot the predicted values for the probability of not developing lung cancer for our model for different years of smoking. We can also distinguish between birdkeepers and non-birdkeepers.

```
# set up axes for plot
plot(c(0,50), c(0,1), type = "n", xlab = "Years smoking",
     ylab = "Prob of No LC")
# create vector with values of YR we want to predict for
yr <- seq(0,50,by=5)
# plot pred values for birdkeepers for 0-50 years smoking
lines(yr,predict(birds.step,data.frame(YR=yr,
     BK=factor(rep("BIRD",length(yr)),levels=BK)),
     type="response"),lty=1)
# plot pred values for non-birdkeepers for 0-50 years smoking
lines(yr,predict(birds.step,data.frame(YR=yr,
     BK=factor(rep("NOBIRD",length(yr)),levels=BK)),
     type="response"),lty=3)
# add legend
legend(0,1,lty=c(1,3),legend=c("BIRD","NOBIRD"))
```

Interpret the plot.

9. We should plot the observed proportions in the above plot as well to judge how well our model is fitting. How can we do this?

```
# compute observed prop of LC for Birdkeepers in age intevals
# 0,[1,20],[21,30],[31,40],[41,50] for years smoked
```

```
prop.0 <- sum(LC[YR==0&BK=="BIRD"]=="NOCANCER")/sum(YR==0&BK=="BIRD")
prop.1to20 <- sum(LC[YR>0&YR<=20&BK=="BIRD"]=="NOCANCER")/
    sum(YR>0&YR<=20&BK=="BIRD")
prop.21to30 <- sum(LC[YR>20&YR<=30&BK=="BIRD"]=="NOCANCER")/
    sum(YR>20&YR<=30&BK=="BIRD")
prop.31to40 <- sum(LC[YR>30&YR<=40&BK=="BIRD"]=="NOCANCER")/
    sum(YR>30&YR<=40&BK=="BIRD")
prop.41to50 <- sum(LC[YR>40&YR<=50&BK=="BIRD"]=="NOCANCER")/
    sum(YR>40&YR<=50&BK=="BIRD")

# Plot observed proportions for birdkeepers with "Y"
points(c(0,10.5,25.5,35.5,45.5),c(prop.0,prop.1to20,
    prop.21to30,prop.31to40,prop.41to50),pch="Y",cex=0.9)

# Repeat above for Non-birdkeepers
prop.0 <- sum(LC[YR==0&BK=="NOBIRD"]==1)/
    sum(YR==0&BK=="NOBIRD")
prop.1to20 <- sum(LC[YR>0&YR<=20&BK=="NOBIRD"]=="NOCANCER")/
    sum(YR>0&YR<=20&BK=="NOBIRD")
prop.21to30 <- sum(LC[YR>20&YR<=30&BK=="NOBIRD"]=="NOCANCER")/
    sum(YR>20&YR<=30&BK=="NOBIRD")
prop.31to40 <- sum(LC[YR>30&YR<=40&BK=="NOBIRD"]=="NOCANCER")/
    sum(YR>30&YR<=40&BK=="NOBIRD")
prop.41to50 <- sum(LC[YR>40&YR<=50&BK=="NOBIRD"]=="NOCANCER")/
    sum(YR>40&YR<=50&BK=="NOBIRD")
# Plot observed proportions for non-birdkeepers with "N"
points(c(0,10.5,25.5,35.5,45.5),c(prop.0,prop.1to20,
    prop.21to30,prop.31to40,prop.41to50),pch="N",cex=0.9)
```

Comment on the fit.
Why should we treat the curve with caution beyond $YR = 50$?