Lab 5 - Classification

The purpose of this lab is to build classifiers for tumor samples using gene expression data and to assess the accuracy of these classifiers. We will use the dataset presented in Golub *et al.* (1999) and available at `http://www.genome.wi.mit.edu/MPR`. It will probably be easier to use the `golubEset` package.
These data come from a study of gene expression in two types of acute leukemias: acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML). Gene expression levels were measured for 47 ALL tumor samples (38 B-cell ALL and 9 T-cell ALL) and 25 AML tumor samples, using Affymetrix high-density oligonucleotide arrays containing $p = 6,817$ human genes. The samples are divided into a training set (LS) of 38 samples and a test set (TS) of 34 samples.
We will work with data that have already been pre-processed as described in Lab 2. You have already done everything in Sections 0 through 4, but will want to repeat some of the filtering to get the data we want to work with.

0. R packages. Load the required R packages for this lab

```
library(cluster)
library(Biobase)
library(annotate)
library(genefilter)
library(golubEsets)
```

1. Expression data. In Lab 2, objects of class `exprSet` were created for the Golub et al. training and test sets. These objects have been saved in the package `golubEsets` and can be accessed using `data(golubTrain)` and `data(golubTest)`. The phenotype slot `phenoData` contains class labels ALL and AML for the samples. Recall that the data were pre-processed as follows, jointly for the training and test set.

(i) thresholding: floor of 100 and ceiling of 16,000;
(ii) filtering: exclusion of genes with $\max / \min \leq 5$ or $(\max - \min) \leq 500$, where $\max$ and $\min$ refer respectively to the maximum and minimum intensities for a particular gene across the mRNA samples;
(iii) base 10 logarithmic transformation.

As a result, expression data on 3571 genes are retained. To view summaries of the expression and tumor class data use `show`.

```
data(golubTrain)
golubTrain
data(golubTest)
golubTest
```

```
##try to subset an exprSet, either by rows or columns
```

For ease of notation, let us store the expression data for the training set in a matrix LS and the tumor class labels in `cl`. Similarly for the test set.

```
LS<-exprs(golubTrain)
cl<-golubTrain$ALL.AML
TS<-exprs(golubTest)
clts<-golubTest$ALL.AML

##mmfilt is in the library

 LS[LS<100]<-100
 TS[TS<100]<-100
 LS[LS>16000]<-16000
 TS[TS>16000]<-16000

 mmfilt <- function(r=5, d=500, na.rm=TRUE) {
     function(x) {
         minval <-  min(x, na.rm=na.rm)
         maxval <-  max(x, na.rm=na.rm)
         (maxval/minval > 5) && (maxval-minval > 500)
     }
 }

 mmfun <- mmfilt()

 ffun <- filterfun(mmfun)

 good <- genefilter(cbind(LS, TS), ffun )

 sum(good) ##I get 3571

 LSsub <- log10(LS[good,])
 TSsub <- log10(TS[good,])
 LTsub <- cbind(LSsub, TSsub)
```

2. Use of annotate.

We now consider the use of the annotate package.

```
ll<- read.annotation("hgu68ll") #locus link
```

```
gname <- row.names(LSsub[good,])[50]
llname<-get(gname, env=ll)
locuslink(llname) ##open your browser to the right thing

chroms <- read.annotation("hgu68Chrom")

whichC <- multiget(row.names(LSsub[good,]), env=chroms, iffail=NA)
cc <-as.numeric(unlist(whichC))
hist(cc)
```

3. Clustering of mRNA samples. Cluster the leukemia mRNA samples using the `hclust` function and varying the number of genes. Do the ALL and AML samples cluster together? Try different between-clusters dissimilarity measures by modifying the "method" argument in `hclust`. Also try different dissimilarities by modifying the "method" argument in `dist`.
Using correlation as similarity function and complete linkage agglomeration

```
gf <- gapFilter(550, 800, .1)
ff <- filterfun(gf)
xx <- 10^LSsub ##to get back to original units
good <- genefilter(xx, gf)
sum(good)  # should be 1093

LS2 <- LSsub[good,]

library(mva)
clust.res<-hclust(as.dist(1-cor(LS2)), method = "complete")
plot.hclust(clust.res, labels=cl,hang = 0.1)
```

Using Euclidean distance and average linkage agglomeration

```
clust.res<-hclust(dist(t(LS2)), method = "average")
plot.hclust(clust.res, labels=cl,hang = 0.1)
```

Optional. Repeat the above analysis with the partitioning clustering algorithm "Partitioning Around Medoids" (PAM) in the `pam` function which is found in the `cluster` library.

```
pamLS <- pam(as.dist(1-cor(LS2)), 3, diss=TRUE)
plot(pamLS)
```

4. Nearest neighbor classifiers.
We are interested in three groups. ALL-B, ALL-T and AML. We need to make up a new variable to discern between these and then we will try some classification.

```
ALL <- c(as.character(golubTrain$ALL),
        as.character(golubTest$ALL))
BT <- c(as.character(golubTrain$T.B),
         as.character(golubTest$T.B))
BTs <- ifelse(BT=="B-cell", "B", "T")
BTs <- ifelse(BT=="T-cell", "T", BTs)
BTs <- ifelse(BT=="NA", "", BTs)
newALL <- factor(paste(ALL, BTs, sep=""))
nALL.train <- newALL[1:38]
nALL.test <-newALL[39:72]

##now do an Anova filter
af <- Anova(nALL.train, p=0.0000001)
ffA <- filterfun(af)
wh.ALL <- genefilter(LSsub, af) #use only training data
sum(wh.ALL)  #how many
```

We now examine k nearest neighbor classifiers. We want to see what happens for different values of k. Genes were selected using the ANOVA filter above. It is also interesting to examine the difference in prediction when the training set is used to predict itself (note that in this case selecting $k = 1$ will give you perfect prediction). We use the function `knn` from `class` package); use `?knn` to see what options are available for this function.
Compare the misclassification rate, i.e., the error rates on the training set. Compare these error rates to test set error rates.

Now build a nearest neighbor classifier with k selected by cross-validation (function `knn.cvk` given below. What is the chosen value of k? Compare resubstitution and test set error rates.

Compare this to a classifier built using all the genes.

```
##how does it work on the test set
pred.test <- knn(t(LSsub[wh.ALL,]), t(TSsub[wh.ALL,]), nALL.train, k=5)
table(pred.test, nALL.test)

##how does it work on the training set
pred.tr <- knn(t(LSsub[wh.ALL,]), t(LSsub[wh.ALL,]), nALL.train, k=5)
table(pred.tr, nALL.train)

##
##use knn and cross-validation to choose the best k!
library(class)
```

```
knn.cvk<-function(LS, cl, k=1:5, l=0, prob=FALSE, use.all=TRUE)
{
  classes<-unique(cl)
  cv.err<-rep(0,length(k))
  cl.pred<-matrix(NA, nrow(LS), length(k))
  for(j in (1:length(k)))
  {
    cl.pred[,j]<-knn.cv(LS, cl, k[j], l, prob, use.all)
    cv.err[j]<-sum(cl!=cl.pred[,j])
   }
  k0<-k[which.min(cv.err)]
  return(k=k0,pred=classes[cl.pred[,which.min(cv.err)]])
}

whichone <- knn.cvk(t(LSsub[wh.ALL,]), nALL.train)
```

You may wish to use the function `plot.classif` to look at partitions produced by the different classifiers (*** works only for bivariate data at the moment).

5. Classification trees. Optional. Repeat 4 with `rpart`.

```
library(rpart)

dorpart <-  function(LS, TS, cl) {
        LS <- as.data.frame(LS)
        TS <- as.data.frame(TS)
        dimnames(TS)[[2]] <- dimnames(LS)[[2]] <- paste("V",
            1:ncol(LS), sep = "")
        fit <- rpart(factor(cl) ~ ., data = LS)
        predict.rpart(fit, TS, type = "class")
    }

all.rp <- dorpart(t(LSsub[wh.ALL,]), t(TSsub[wh.ALL,]), nALL.train)
table(all.rp, nALL.test)
```

6. Discriminant analysis. Optional. Repeat 4 with `lda`.

```
dolda <- function(LS, TS, cl, prior, pmethod="plug-in") {
```

```
     ndata <- data.frame(LS)
     ndata$cl <- cl
     if( missing(prior) dolda <- function(LS, TS, cl, prior, pmethod="plug-i
     ndata <- data.frame(LS)
     ndata$cl <- cl
     if( missing(prior) )
         mn <- lda(cl~., data=ndata)
     else
         mn <- lda(cl~., data=ndata, prior=prior)
     pdata<-data.frame(TS)
     preds <- predict(mn, pdata, method=pmethod)
     rval <- list(lda = mn, preds=preds)
     class(rval) <- "mylda"
     rval
 }

lda.test <- dolda(t(LSsub[wh.ALL,]), t(TSsub[wh.ALL,]), nALL.train)
table(lda.test$preds$class, nALL.test)

##how does it work on the training set
##not very meaningful
lda.pred.tr <- dolda(t(LSsub[wh.ALL,]), t(LSsub[wh.ALL,]), nALL.train)
table(lda.pred.tr$preds$class, nALL.train)
)
         mn <- lda(cl~., data=ndata)
     else
         mn <- lda(cl~., data=ndata, prior=prior)
     pdata<-data.frame(TS)
     preds <- predict(mn, pdata, method=pmethod)
     rval <- list(lda = mn, preds=preds)
     class(rval) <- "mylda"
     rval
 }

lda.test <- dolda(t(LSsub[wh.ALL,]), t(TSsub[wh.ALL,]), nALL.train)
table(lda.test$preds$class, nALL.test)

##how does it work on the training set
##not very meaningful
lda.pred.tr <- dolda(t(LSsub[wh.ALL,]), t(LSsub[wh.ALL,]), nALL.train)
table(lda.pred.tr$preds$class, nALL.train)
```

7. Bagging.

Use the `agg.class` function from the `AggPred` package to bag a classifier of your choice (e.g.

knn, lda, qda, or rpart). Note, this package is still very experimental and will change in the near future.

At each bagging iteration, select only genes which have a nominal p-value less than 0.000001 for the ANOVA filter applied to the bagged training set.

What is the out-of-bag error rate? Plot the prediction votes and vote margins for the learning set. How do the prediction votes compare to the vote margins? What is the test set error rate? Plot the prediction votes.