Practical DNA Microarray Analysis
Heidelberg, Oct 2003

Model Assessment and Selection: Exercises

Axel Benner
Biostatistik, DKFZ, D-69120 Heidelberg, Germany

Introduction

The exercises provided here are of a general form and may be applied to the data sets `BootStrap.RData` or `breastcancer.Rdata` for model assessment.

It may be useful to start with a clean workspace. If you want to do so, then remove (almost) everything in the working environment by

```
rm(list=ls())
```

Load new data, e.g. the expression set used in the bootstrapping exercise, containing microarrays of 52 women with breast cancer of whom 34 did not experience a recurrence of the tumour during a 3 years time period.

```
load("BootStrap.Rdata")
require(affy)
```

1 K-fold adjusted cross-validation

Algorithm 1 shows an algorithm to perform K-fold adjusted cross-validation for an arbitrary regression model.
We will implement the algorithm in *R* and apply it to a logistic regression model using a single gene as predictor which can be extended to the whole gene expression set by penalized logistic regression (which will not be discussed her). The probe set no. 4402 with Affymetrix ID “34361_at” will be chosen.

```
gene.exprs <- exprs(Huang.RE)[4402,]
# get the response vector
rec.info <- pData(Huang.RE)$Recurrence
```

Algorithm 1 K-fold adjusted cross-validation

```
1. Fit the regression model to all cases, calculate predictions $\hat{y}_i$ from that model, and
   average the values of $c(y_i, \hat{y}_i)$ to get $D$.
2. Choose group sizes $m_1, ..., m_K$ such that $m_1 + ... + m_K = n$.
3. For $k = 1, ..., K$
   (a) choose $C_k$ by sampling $m_k$ times without replacement from {1,2,...,n} minus
       elements chosen for previous $C_i$s
   (b) fit the regression model to all data except cases $i \in C_k$
   (c) calculate new predictions $\hat{y}_i = \mu(x_i, \hat{F}_{-k})$ for $i \in C_k$
   (d) calculate predictions $\hat{y}_{ki} = \mu(x_i, \hat{F}_{-k})$ for all $i$; then
   (e) average the $n$ values $c(y_i, \hat{y}_{ki})$ to give $D(\hat{F}, \hat{F}_{-k})$.
4. Average the $n$ values of $c(y_i, \hat{y}_i)$ using $\hat{y}_i$ from step 3(c) to give $\hat{\Delta}_{cv,K}$.
5. Calculate $\Delta_{acv,K} = \Delta_{cv,K} + D(\hat{F}, \hat{F}) - \sum_{k=1}^{K} p_k D(\hat{F}, \hat{F}_{-k})$ with $p_k = m_k/n$.
```

1.1 Implementing the algorithm

Step 1. Fit the logistic regression model using function `glm`

```
fit.all <- glm(rec.info ~ gene.exprs, family = binomial)
pred.all <- ifelse( predict(fit.all, type="response") < 0.5, 0, 1)
```

Step 2. Choose $k = min(\sqrt{52}, 10) = 7$ disjoined subgroups using function `balanced.folds` from the `pamr` package

```
require(pamr)
set.seed(54321)
n<-length(gene.exprs)
k<-floor(min(c(sqrt(n),10)))
cc <- balanced.folds(rec.info, nfold=k)
```

Step 3. Compute $D(\hat{F}, \hat{F}_{-k})$ for each k using the misclassification loss function

$$c(y_+, \hat{y}_+) \;=\; \begin{cases} 1, & y_+ \neq \hat{y}_+ \\ 0, & \text{otherwise} \end{cases}$$

```
pred.c <- list()
error.i <- vector(length=k)

for (i in 1:k)
{
# 3(a)(b)
fit.i <- glm(rec.info ~ gene.exprs, subset=-cc[[i]], family = binomial)
# 3(c)
pred.i <- ifelse(predict(fit.i, newdata=data.frame(gene.exprs=gene.exprs[cc[[i]]]),
          type='response') < 0.5, 0, 1)
pred.c[[i]] <- pred.i
# 3(d)
pred.all.i <- ifelse(predict(fit.i,newdata=data.frame(gene.exprs=gene.exprs),
          type='response') < 0.5, 0, 1)
# 3(e)
error.i[i] <- sum(rec.info!=pred.all.i)/n
}
```

Step 4. Compute $\hat{\Delta}_{cv,K}$

```
pred.cc <- cbind(unlist(cc), unlist(pred.c))
delta.cv.k <- sum(rec.info!=pred.cc[order(pred.cc[,1]),2])/n
```

Step 5. Compute $\hat{\Delta}_{acv,K}$

```
p.k <- unlist(lapply(cc, length))/n
# delta.app: apparent error
delta.app <- mean(rec.info!=pred.all)

delta.acv.k <- delta.cv.k + delta.app - sum(p.k*error.i)

print(delta.acv.k)
```

```
[1] 0.2067308
```

Note that we have selected a single probe set using prior information. Including the gene selection process into the computation of the prediction error this probably result in a much higher prediction error.

2 Estimation of the prediction error

For simplicity we select the 1000 most variable probe sets (selected by using the coefficient of variation) for this exercise (which will be stored in the data frame `mydata`),

```
sd.exp <- apply(exprs(Huang.RE),1,sd)
mean.exp <- apply(exprs(Huang.RE),1,mean)

cv.exp <- sd.exp/mean.exp
index <- order(cv.exp,decreasing=TRUE)[1:1000]

mydata <- data.frame(t(exprs(Huang.RE)[index,]),
               Recurrence=as.factor(pData(Huang.RE)$Recurrence))
```

The function `errorest` implements a unified interface to several resampling based estimators. One specifies the error rate estimator by `estimator = "cv", "boot"` or `"632plus"`, respectively. A 10-fold cross validation is performed by choosing `estimator = "cv"` and `est.para = control.errorest(k = 10)`. The options `estimator = "boot"` or `estimator = "632plus"` deliver a bootstrap estimator and its bias corrected version *.632+*. We can specify the number of bootstrap samples to be drawn by `est.para = control.errorest(nboot = 25)`.

The argument `predict` represents the chosen predictor function. For a unified interface `predict` has to be based on the arguments `object` and `newdata` only. Therefore a wrapper function `mypredict` is necessary for classifiers which require more than those arguments or do not return the predicted classes by default. For a linear discriminant analysis with `lda`, we need to specify

```
mypredict.lda <- function(object, newdata) {
     predict(object, newdata = newdata)$class
 }
```

and calculate a 10-fold-cross-validated error rate estimator for a linear discriminant analysis of the subset of the 1000 most variable probe sets (i.e. `newdata`) by calling

```
require(ipred)
errorest(Recurrence ~ ., data = mydata,
       model = lda, estimator = "cv", predict = mypredict.lda)
```

Often it may be useful to reduce the number of predictors/genes before training a classifier. Estimating the error rate after the variable selection leads to biased estimates of the misclassification error and therefore one has to estimate the error rate of the whole procedure. Within the `errorest` framework, this can be easily done.
First, we define a function which does both variable selection and training of the classifier. As an example, we select the predictors by comparing their univariate p-values of a two-sample t-test with a pre-specified level and train a LDA using the selected variables only.

```
mymod <- function(formula, data, level = 0.05) {
  sel <- which(lapply(data, function(x) {
     if (!is.numeric(x))
          return(1)
     else return(t.test(x ~ data$Recurrence)$p.value)
  }) < level)
  sel <- c(which(colnames(data) %in% "Recurrence"), sel)
  mod <- lda(formula, data = data[, sel])
  function(newdata) {
  predict(mod, newdata = newdata[, sel])$class
 }
}
```

Note that `mymod` does not return an object of class `lda` but a function with argument `newdata` only.

Computing a 7-fold cross-validated error rate estimator is then done by

```
set.seed(71003)
errorest(Recurrence ~ ., data=mydata, model=mymod, est.para=control.errorest(k=7))

Call:
errorest.data.frame(formula = Recurrence ~ ., data = mydata,
    model = mymod, est.para = control.errorest(k = 7))

         7-fold cross-validation estimator of misclassification error

Misclassification error:  0.3077
```

2.1 Computing different error rate estimators

Leave-one-out cross-validation

```
set.seed(71003)
errorest(Recurrence ~ ., data=mydata, model=mymod, est.para=control.errorest(k=52))

Call:
errorest.data.frame(formula = Recurrence ~ ., data = mydata,
    model = mymod, est.para = control.errorest(k = 52))

         52-fold cross-validation estimator of misclassification error

Misclassification error:  0.2308
```

Bootstrap (B=10)

```
set.seed(71003)
errorest(Recurrence ~ . ,data=mydata, model=mymod,estimator="boot",
                         est.para=control.errorest(nboot = 10))

Call:
errorest.data.frame(formula = Recurrence ~ ., data = mydata,
    model = mymod, estimator = "boot", est.para = control.errorest(nboot = 10))

         Bootstrap estimator of misclassification error
         with 10 bootstrap replications

Misclassification error:  0.2948
Standard deviation: 0.0593
```

Bootstrap (B=20)

```
set.seed(71003)
errorest(Recurrence ~ . ,data=mydata, model=mymod,estimator="boot",
                         est.para=control.errorest(nboot = 20))
```

```
Call:
errorest.data.frame(formula = Recurrence ~ ., data = mydata,
    model = mymod, estimator = "boot", est.para = control.errorest(nboot = 20))

         Bootstrap estimator of misclassification error
         with 20 bootstrap replications

Misclassification error:  0.3308
Standard deviation: 0.0328
```

.632+ Bootstrap (B=10)

```
set.seed(71003)
errorest(Recurrence ~ . , data=mydata, model=mymod,estimator="632plus",
                          est.para=control.errorest(nboot = 10))
```

```
Call:
errorest.data.frame(formula = Recurrence ~ ., data = mydata,
    model = mymod, estimator = "632plus", est.para = control.errorest(nboot = 10))

         .632+ Bootstrap estimator of misclassification error
         with 10 bootstrap replications

Misclassification error:  0.2501
```

.632+ Bootstrap (B=25)

```
set.seed(71003)
errorest(Recurrence ~ . , data=mydata, model=mymod,estimator="632plus",
                          est.para=control.errorest(nboot = 25))
```

```
errorest.data.frame(formula = Recurrence ~ ., data = mydata,
    model = mymod, estimator = "632plus", est.para = control.errorest(nboot = 25))

         .632+ Bootstrap estimator of misclassification error
         with 25 bootstrap replications

Misclassification error:  0.2862
```

The smallest prediction error estimate result from using the leave-one-out cross-validation. Does this mean, that this is the best estimation technique for these data?

Change the inclusion level of the t-test selector to 0.01. How does this change the resulting estimates of the prediction error?

3 Estrogen receptor status data

Use the data from the molecular diagnosis exercise

```
load("breastcancer.RData")
```

which consists of 46 breast tumor samples where 23 samples were positive for estrogen receptor (class ER+) and 23 were negative (class ER-).

Try to repeat the estimation of the prediction error for this data.

4 Additional information

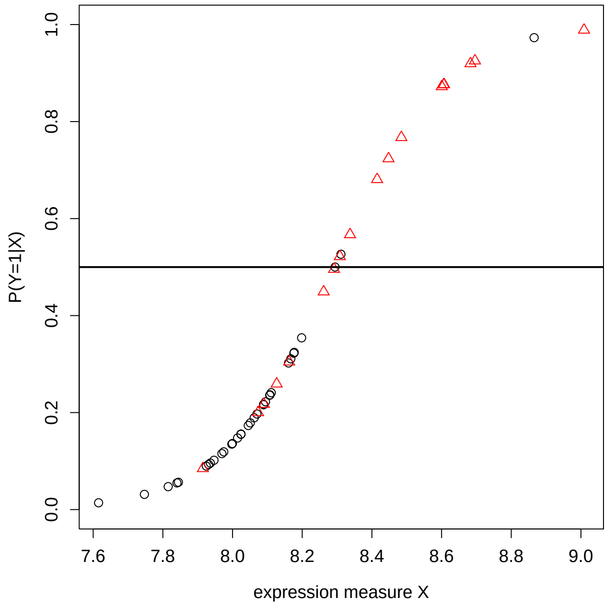

Figure 1: Logistic regression fit with respect to probe set 34361_at applied to the tumor recurrence data. The 18 recurrent cases are marked by red triangles.

Binary logistic regression

Having a vector of predictors $X = (X_1, \dots, X_p)^T$ and a binary response Y with Y equal to 1 if the event of interest occurred, and zero otherwise, the binary logistic regression model is defined by

$$P(Y = 1|\mathbf{X}) = \frac{1}{1 + \exp\{-\mathbf{X}\boldsymbol{\beta}\}}$$

with $\mathbf{X}\boldsymbol{\beta} = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$, and parameter vector $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$. The regression parameters $\boldsymbol{\beta}$ are estimated by the method of maximum likelihood. The function $f(x) = [1 + \exp\{x\}]^{-1}$ is called the logistic function. The logit transformation of $P(Y = 1)$ is linear in $\mathbf{X}\boldsymbol{\beta}$

$$\text{logit}(Y = 1|\mathbf{X}) = \text{logit}(p) = \log \frac{p}{1-p} = \mathbf{X}\boldsymbol{\beta}$$

with $p = P(Y = 1|\mathbf{X})$.

Fisher's Linear Discriminant Analysis

The task is to discriminate two classes using data $\mathbf{X}_1 = (\mathbf{x}_{11}, \ldots, \mathbf{x}_{\mathbf{1n_1}}) \in I\!R^{p \times n_1}$ in class 1 and $\mathbf{X}_2 = (\mathbf{x}_{21}, \ldots, \mathbf{x}_{\mathbf{2n_2}}) \in I\!R^{p \times n_2}$ in class 2 by reducing the p-dimensional problem to a one-dimensional one by the transformation

$$y_{ki} = \mathbf{a}^T \mathbf{x}_{ki}, \quad \mathbf{a} = (a_1, \ldots, a_p)^T, \; i = 1, \ldots, n_k, \; k = 1, 2$$

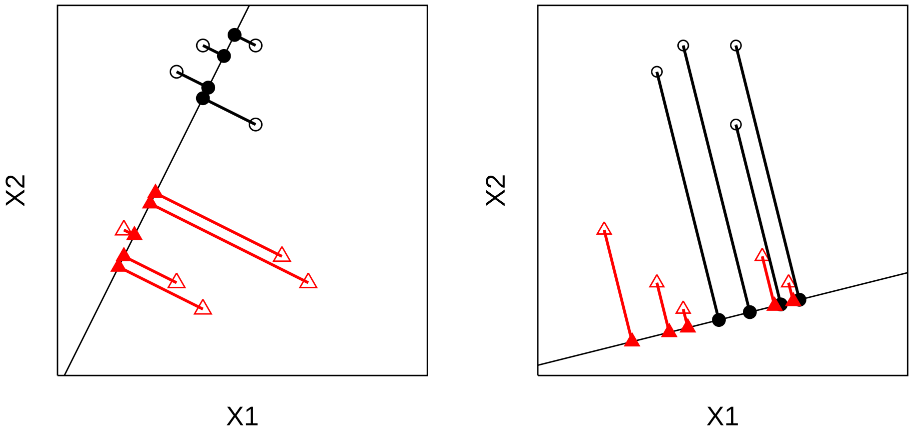

Figure 2: Idea of the linear discriminant analysis. The left panel shows a direction which separates perfectly, whereas the right panel shows a direction where a perfect discrimination is not possible.

The linear combination $\mathbf{a}^T\mathbf{x}$ corresponds for $\|\mathbf{a}\| = 1$ to the geometric projection of $\mathbf{x}$ onto a line through the origin with direction vector $\mathbf{a}$.
With $\overline{y}_k = \mathbf{a}^T\overline{\mathbf{x}}_k$, $k = 1, 2$ arithmetic means of the two classes we get the sums of squared deviations as

$$s_k^2 = \sum_{i=1}^{n_k} (y_{ki} - \overline{y}_k)^2$$

The criterion used is the deviation of the y-means relative to the overall sum of squared deviations

$$Q(\mathbf{a}) = \frac{(\overline{y}_1 - \overline{y}_2)^2}{s_1^2 + s_2^2} \; .$$

It holds $s_1^2 + s_2^2 = \mathbf{a}^T\mathbf{W}\mathbf{a}$ with

$$\mathbf{W} = (n_1 + n_2 - 2)\mathbf{S} = \sum_{i=1}^{n_1} (\mathbf{x}_{1i} - \overline{\mathbf{x}}_1)(\mathbf{x}_{1i} - \overline{\mathbf{x}}_1)^T + \sum_{i=1}^{n_2} (\mathbf{x}_{2i} - \overline{\mathbf{x}}_2)(\mathbf{x}_{2i} - \overline{\mathbf{x}}_2)^T$$

($\mathbf{S}$ = pooled empirical covariance matrix)
and so we have to solve the maximization problem

$$Q(\mathbf{a}) = \frac{(\mathbf{a}^T\overline{\mathbf{x}}_1 - \mathbf{a}^T\overline{\mathbf{x}}_2)^2}{\mathbf{a}^T\mathbf{W}\mathbf{a}} \rightarrow \max_{\mathbf{a}\neq\mathbf{0}}$$

The normal equations are

$$\frac{\partial Q(\mathbf{a})}{\partial \mathbf{a}} = \frac{2(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)\mathbf{a}^T\mathbf{W}\mathbf{a} - 2\mathbf{W}\mathbf{a}(\mathbf{a}^T\overline{\mathbf{x}}_1 - \mathbf{a}^T\overline{\mathbf{x}}_2)}{(\mathbf{a}^T\mathbf{W}\mathbf{a})^2} = \mathbf{0}$$

$$\Rightarrow \quad \overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2 = \mathbf{W}\mathbf{a}\left(\frac{\mathbf{a}^T\overline{\mathbf{x}}_1 - \mathbf{a}^T\overline{\mathbf{x}}_2}{\mathbf{a}^T\mathbf{W}\mathbf{a}}\right) \quad \Rightarrow \quad \mathbf{a} = \frac{\mathbf{W}^{-1}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)}{\|\mathbf{W}^{-1}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)\|}$$

The classification rule for a new object $\mathbf{x}$ is then:

Compute $y = \mathbf{a}^T\mathbf{x}$ and assign $\mathbf{x}$ to class 1 if y is closer to $\overline{y}_1$ than to $\overline{y}_2$,
i.e. $|y - \overline{y}_1| < |y - \overline{y}_2| \Leftrightarrow y > \frac{1}{2}(\overline{y}_1 + \overline{y}_2)$ or

$$\mathbf{a}^T(\mathbf{x} - \frac{1}{2}(\overline{\mathbf{x}}_1 + \overline{\mathbf{x}}_2)) > 0$$