1

Computational Inference

Torsten Hothorn, Marcel Dettling, Peter Bühlmann

1.1 Introduction

Recent technological developments have opened new perspectives in biomedical research and clinical practice, and they also changed the way in which statistics makes progress. The main challenge is to develop computational inference tools that can deal with very high dimensional datasets containing thousands of input variables for a few dozens of experiments only. Such data structures do emerge for example from the gene expression microarray technology, and from protein mass spectroscopy analysis.

In this chapter, we focus on tumor prognosis with gene expression data. Given efficient data analysis tools, information from biotechnology may represent a promising supplement to tumor prediction based on traditional clinical factors. Let us assume that we are given previous experience from a cohort of I patients, which mathematically amounts to a learning sample

$$\mathcal{L} = \{(y_1, x_1), (y_2, x_2), \dots, (y_I, x_I)\}.$$

The input variable $x_i \in \mathbb{R}^J$ contains the vector-valued gene expression profile of J genes for the ith patient. The response variable can be binary with $y_i \in \{0, 1\}$ coding for two different populations or classes, as for example different cancers or tumor subtypes. For notational convenience, multiclass prediction and the analysis of survival data are to be discussed later in Sections 1.4 and 1.7

We plan to systematically exploit the information in the learning sample $\mathcal{L}$ for defining a rule that predicts the disease evolvement (the response variable) from the gene expression profile x. This rule can then be applied to novel cancer patients for establishing an early and precise prognosis, which is often crucial for a treatment with little side-effects and good cure rates. From a mathematical viewpoint, this amounts to learning a function $g : \mathbb{R}^J \to \mathbb{R}$ which can be applied in order to predict the class membership of an observation with expression profile x via some fixed transformation f. For example, $g(x|\mathcal{L}) = \log(\hat{p}(x)/(1 - \hat{P}(x)))$, where $\hat{p}(x) = \widehat{Pr}[Y = 1|X = x]$ and $f(g) = \chi(g > 0)$ which yields a classification rule respecting equal misclassification costs, where χ denotes the indicator function. In the sequel, such procedures are referred to as classifiers. Although classification is a well-known methodology in statistics, finding solutions to problems with many input variables or genes J but a very limited number of patients I is challenging. A promising approach that is suitable for high-dimensional prediction problems in bioinformatics is to use ensemble methods that aggregate many simple classifiers into a powerful committee.

1.1.1 Ensemble Methods

In contrast to using just a single fit from a particular method, ensemble techniques aim at improving the predictive ability of a relatively simple statistical learning technique by constructing linear combinations thereof. Technically, the ensemble is written as

$$g_{E(M)}(\cdot|\mathcal{L}) = \sum_{m=1}^{M} \alpha_m g_m(\cdot|\mathcal{L}_m),$$

where $\mathcal{L}_m$ is a reweighted version of the learning sample $\mathcal{L}$ and α_m are real valued weights. Here, g_m is a base procedure, which we term the 'weak learner'. For example, $g_m(x)$ may be a simple estimate $\log(\hat{p}(x)/(1-\hat{p}(x)))$ of the log-odds ratio. The choice of the reweighted sample $\mathcal{L}_m$, the aggregation weights α_m, as well as of the weak learner are the art of this business. Bagging, boosting and random forests are such implementations, which will be discussed below. Apparently, the final ensemble estimator depends on the choice of the weak learner. The most prominent choice in high-dimensional problems are recursive partitioning methods ('classification and regression trees'), as they incorporate some form of feature selection and tend to work well when there are more input variables than there are observations.

We refer to Breiman et al. [1] for a detailed description on recursive partitioning methods. The *rpart* package [2] essentially implements the methodology described by Breiman et al. [1].

1.2 Bagging & Random Forests

Bagging [3] is a rather simple but effective ensemble method. Its name is an acronym for **b**ootstrap **agg**egat**ing** which suggests the principle idea of this procedure: Predictions of weak learners fitted to bootstrap samples $\mathcal{L}_m$ of the original learning sample $\mathcal{L}$ are aggregated by a majority vote.

A bootstrap sample from the original learning sample $\mathcal{L}$ is an i.i.d. random sample of I observations $(y_i^*, x_i^*), i = 1, \dots, I$, where the probability of selecting observation (y_i^*, x_i^*) from $\mathcal{L}$ is $1/I$. A bootstrap sample is thus a sample from the empirical distribution function of the learning sample. The algorithm works as follows.

1. Draw M bootstrap samples $\mathcal{L}_m, m = 1, \dots M$, from the original learning sample $\mathcal{L}$.
2. Fit a weak learner for each of the bootstrap samples $g_m(\cdot|\mathcal{L}_m)$, and construct the classifiers $f(g_m(\cdot|\mathcal{L}_m))$.
3. Aggregate the classifiers using weights $\alpha_m = 1/M$, yielding the ensemble
$$g_{E(M)}(\cdot|\mathcal{L}) = M^{-1} \sum_{m=1}^{M} f(g_m(\cdot|\mathcal{L}_m)).$$

Thus, the ensemble yields the fraction of weak learners predicting class 1. The bagging ensemble votes for class 1 if the majority of the M weak learners votes for 1; and vice versa. More generally, the ensemble predicts class 1 when the fraction of weak learners predicting class 1 exceeds some number ν.

It has been argued that for unstable estimation methods such as decision trees, bagging reduces the variance while the bias remains approximately the same [4].

It is rather straightforward to implement the bagging procedure in high-level languages like S. The basic ingredient is a tree building algorithm used for fitting trees to bootstrap samples of the learning sample.

```
R> simple_bagging <- function(x, lsample, M = 100,
+     nu = 0.5) {
+     I <- nrow(lsample)
+     bsample <- rmultinom(M, I, rep(1, I)/I)
+     pred <- rep(0, nrow(x))
+     for (m in 1:M) {
+         weaktree <- rpart(y ~ ., data = lsample,
+             weights = bsample[, m], control = rpart.control(xval = 0,
+                 cp = 0.01))
+         prtree <- predict(weaktree, newdata = x,
+             type = "class")
+         pred <- pred + (prtree == levels(lsample$y)[2])
+     }
+     factor(pred/M > nu, levels = levels(lsample$y))
+ }
```

First, M random index vectors representing M bootstrap samples are drawn from the multinomial distribution. A classification tree is fitted to the data using `weights = bsample[,m]` representing the mth bootstrap sample. Large trees without applying any form of pruning whatsoever are grown. The vector `pred` of length I counting the number of trees predicting the second class is updated. Finally, the function returns a factor coding the ensemble predictions obtained from a simple majority vote.

A simple but extremely successful modification to this algorithm is the random forest approach [5]. The basic idea is to modify the tree growing algorithm leading to even weaker components of the ensemble. This is achieved by choosing a small random subset of inputs available for splitting at each stage of the recursive partitioning algorithm building the tree. Thus, the input variables actually used in each of the weak learners are, to a large extent, determined at random.

1.3 Boosting

Boosting has been introduced to the machine learning literature by Freund and Schapire [6] and has demonstrated empirical success on a wide variety of especially high-dimensional prediction problems. The initial notion was that in each boosting iteration, the cases that were misclassified in the previous round get their weights increased, whereas the weights are decreased for cases that were correctly classified. Thus, unlike in bagging and random forests, both the aggregation weights α_m and the reweighted learning samples $\mathcal{L}_m$ depend on the previous function fits $g_1(\cdot|\mathcal{L}_1), \dots, g_{m-1}(\cdot|\mathcal{L}_{m-1})$. However, rather than as a sequential data reweighting scheme, boosting can more fruitfully be seen as a forward stagewise strategy, working by iterative optimization of an empirical risk function

$$R(\mathcal{L}, p(x), L) = \frac{1}{I} \sum_{i=1}^{I} L(y_i, p(x_i)), \; p(x) = Pr[Y = 1|X = x],$$

from the learning set $\mathcal{L}$ via constrained (imposed by the weak learner) functional gradient descent, where $L(\cdot, \cdot)$ is a statistically motivated loss function. If we employ the binomial log-likelihood

$$L(y, p(x)) = y \cdot \log(p(x)) + (1 - y) \cdot (1 - p(x)),$$

a continuous surrogate for the 0/1-misclassification loss and a very established criterion for binary classification, it has been shown that the resulting logitboost algorithm [7] yields an approximation to half of the log-odds ratio. That is,

$$g_{E(M)}(x|\mathcal{L}) = \sum_{m=1}^{M} \alpha_m g_m(x|\mathcal{L}_m) \approx \frac{1}{2} \log \left(\frac{p(x)}{1 - p(x)} \right).$$

Hence, logitboost is a linear expansion in terms of of weak learners $g_m(\cdot|\mathcal{L}_m)$ on the logit scale, constructed by stagewise optimization of the binomial log-likelihood. Estimated conditional class probabilities are obtained by the simple transformation

$$p(x) = \frac{1}{1 + \exp(-2g_{E(M)}(x))},$$

which can be used for class prediction, using a threshold that depends on the misclassification costs. Logitboost has been demonstrated to be a competitive prediction algorithm for tumor classification with microarray data [8, 9]. The procedure works as follows.

1. Initialize $p(x_i) \equiv 1/2, i = 1, \dots, I; m = 1; g_{E(0)}(\cdot|\mathcal{L}) \equiv 0$.
2. Build the pseudo-response for each observation i

   $$\begin{aligned} w_i &= p(x_i)(1 - p(x_i)) \\ u_i &= \frac{y_i - p(x_i)}{w_i}, \end{aligned}$$

   setup-up the new learning sample

   $$\mathcal{L}_m = \{(u_i, x_i); i = 1, \dots, I\}$$

   and fit the weak learner $g_m(\cdot|\mathcal{L}_m)$ with case weights w_i.
3. Update ensemble $g_{E(m)}(\cdot|\mathcal{L}) = g_{E(m-1)}(\cdot|\mathcal{L}) + \frac{1}{2}g_m(\cdot|\mathcal{L}_m)$ and $p(x_i) = 1/(1 + \exp(-2g_{E(m)}(x_i|\mathcal{L})))$
4. repeat until $m = M$.

Predictions are computed with $\alpha_m = 1/2$ and $f(z) = \chi(\exp(1 - 2z) > \frac{1}{2})$.

The definition of the weights w_i in the logitboost algorithm is such that each weak learner is forced to focus on observations close to the decision boundary, i.e., data points where the boosting classifier is in doubt about the predicted class. The final number of boosting iterations M regulates the complexity of the prediction model, early stopping is a form of shrinkage. In the context of microarray data, we recommend a default value of $M = 100$, which is a reasonable compromise between computing time, predictive accuracy and prevention of overfitting. This choice was shown to be empirically superior to approaches where M was estimated on the training data via cross validation [8]. Provided that an interface to the weak learning algorithm is present, an implementation of boosting in high-level languages like S is straightforward and simply consists of a loop that incorporate all the updating operations and the weak learner fit.

1.4 Multiclass Problems

Since there are often no genes that accurately discriminate multiple classes, we recommend to split K-class problems into multiple binary ones. In the context of microarray data, we have collected some empirical evidence that this is more accurate than simultaneous multiclass approaches [8]. The simplest solution is the *one-against-all* approach, which works by defining the response in the kth problem as $y^{(k)} = 1$ if $y = k$, and $y^{(k)} = 0$ else. Then, we are running boosting K times on the modified data $\mathcal{L}^{(k)} = \{(y_1^{(k)}, x_1), \dots, (y_I^{(k)}, x_I)\}$. The estimated conditional class probabilities are normalized and can in turn be used for maximum likelihood classification via

$$\begin{aligned} \widehat{p}^{(k)}(x) &= \frac{\widehat{Pr}_{\mathcal{L}^{(k)}}(y^{(k)} = 1|x)}{\sum\limits_{k=0}^{K-1} \widehat{Pr}_{\mathcal{L}^{(k)}}(y^{(k)} = 1|x)} \\ \widehat{y}(x) &= \underset{k \in \{0,\dots,K-1\}}{\text{argmax}} \; \widehat{p}^{(k)}(x) \\ . \end{aligned}$$

Depending on the data, other schemes than *one-against-all* may be more accurate for splitting polytomous into multiple binary problems.

Note that no additional procedures are necessary in order to deal with multiclass responses for bagging trees or random forests: Estimated conditional class probabilities arise directly from the ensemble of trees.

1.5 Evaluation

The choice of an appropriate classifier for a prediction problem at hand is by no means obvious. Two problems can be separated: The method selection and the error rate estimation tasks. The first task is concerned with choosing the best method available from a, possible huge, set of statistical procedures capable to deal with the problem. For the second one, we try to come up with a realistic assessment of the prediction error of the selected procedure. This information is extremely important when we need to take the decision whether it is worth or even ethical to apply a certain classifier in realistic setups.

The prediction error is measured by a scalar loss function $L(y, \hat{y})$ assessing the goodness of the prediction $\hat{y}$ for some response y. When the response is a categorical variable with classes $\{0, \dots, K-1\}$, the misclassification error $L(y, \hat{y}) = \chi(y \neq \hat{y})$ is an often used loss function. However, this choice is not necessarily the one we are interested in. When we are faced with a two class problem aiming at predicting whether a person suffers a rare but dangerous disease or not the loss of missing an affected person is much higher compared to the loss induced by a false positive detection. For such problems, the misclassification loss is of the form $L(y, \hat{y}) = c_1\chi(y = 0, \hat{y} = 1) + c_2\chi(y = 1, \hat{y} = 0)$ for some misclassification costs c_1, c_2. We may be also interested in measuring the accuracy of the estimated probability $\hat{p}(x)$: Then, the negative log-likelihood can serve as a useful loss function.

A major problem is that the learning sample used for model fitting can not be used for the estimation of the prediction error of that model. Such an estimate would be optimistically biased because the same observations were used for model building and evaluation. Resampling methods like the bootstrap or cross-validation have been studied extensively, a practical introduction can be found in [10]. Here, we will use the notion of out-of-bag estimation [11]. When a bootstrap sample of the original learning sample $\mathcal{L}$ is drawn some observations are left out due to sampling *with* replacement. Those observations can be used as independent sample for the assessment of the prediction error. We can draw random samples from the distribution of the prediction error as follows.

1. Draw B bootstrap samples $\mathcal{L}_b, b = 1, \dots B$, from the original learning sample $\mathcal{L}$.
2. Fit a model to each bootstrap sample $\mathcal{L}_b$, i.e. $g(\cdot|\mathcal{L}_b)$ and assess the error p_b by the loss averaged over the predictions of the observations in the out-of-bootstrap sample

$$p_b = |\mathcal{L} \setminus \mathcal{L}_b|^{-1} \sum_{(y,x) \in \mathcal{L} \setminus \mathcal{L}_b} L(f(g(x|\mathcal{L}_b))), y), \; b = 1, \dots, B.$$

The B error rates can now be visualized, for example by means of boxplots or can be described via estimates of certain parameters such as mean or variance, say. When multiple candidates models are under consideration, we obtain a sample of B error rates for each of them: Those distributions can be compared in order to identify the best or a set of the best algorithms. The null-hypothesis of equality of the bootstrap-distributions of the prediction error of several algorithms can be tested, for example by means of the Friedman test. It should be noted that a rejection of this hypothesis should not be generalized to the population from which the data was drawn, since the inference is conditional on the given learning sample. All what we can conclude is whether a finite number B of bootstrap replicates is sufficient for detecting performance differences in the exact bootstrap distributions, with $B = \infty$ (or from the typically infeasible exact multinomial distribution induced by the bootstrap), of different algorithms. Theoretical justification and illustration of this approach can be found elsewhere [12].

1.6 Applications: Tumor Prediction

1.6.1 Acute Lymphoblastic Leukemia

In this first example, we apply ensemble methods to construct a model that regresses the stage of acute lymphoblastic leukemia (ALL) on the microarray expression levels. We are primarily interested in two questions. At the one hand, we would like to investigate if there is any information about the stage of the disease covered by the microarray expression levels, i.e., a test of the null-hypothesis that the stage of the disease is independent of the expression levels measured. If we are able to reject this global null-hypothesis, we want to build a model that allows us to predict the stage of the disease of a patient based on the microarray expression levels only. Data from patients suffering from both T- and B-cell leukemia are available from the study of Chiaretti et al. [13], and we restrict ourselves to patients with B-cell leukemia.

reference to chapter?

The package *ALL* offers a data object `ALL`, an object of class *exprSet*, which contains the data of patients suffering from both T- and B-cell leukemia.

```
R> pkgload <- require(Biobase, quietly = TRUE)

Welcome to Bioconductor
         Vignettes contain introductory material.  To view,
         simply type: openVignette()
         For details on reading vignettes, see
         the openVignette help page.

R> pkgload <- require(ALL, quietly = TRUE)
R> data(ALL)
```

We are provided with expression levels of 12625 genes for 128 patients. For the analysis here, we restrict the data to patients suffering B-cell leukemia, with 5 subclasses, and to expression levels of genes with coefficient of variation between 0.08 and 0.18.

```
R> cvv <- apply(exprs(ALL), 1, function(x) sd(x)/mean(x))
R> ok <- cvv > 0.08 & cvv < 0.18
R> BStagelev <- paste("B", c("", 1:4), sep = "")
R> BALL <- ALL[ok, ALL$BT %in% BStagelev]
R> pData(phenoData(BALL))$BStage <- factor(BALL$BT,
+     levels = BStagelev)
```

The class distribution of the stages of the disease (`BALL$BStage`) can be inspected via

```
R> table(BALL$BStage)

 B B1 B2 B3 B4
 5 19 36 23 12
```

and is depicted by means of a barplot in Figure 1.1.

Although we selected 3841 genes showing a reasonable variation among the patients, some mild form of univariate variable selection will help to circumvent computational difficulties. Here, we measure the association

```
R> barplot(table(BALL$BStage))
```

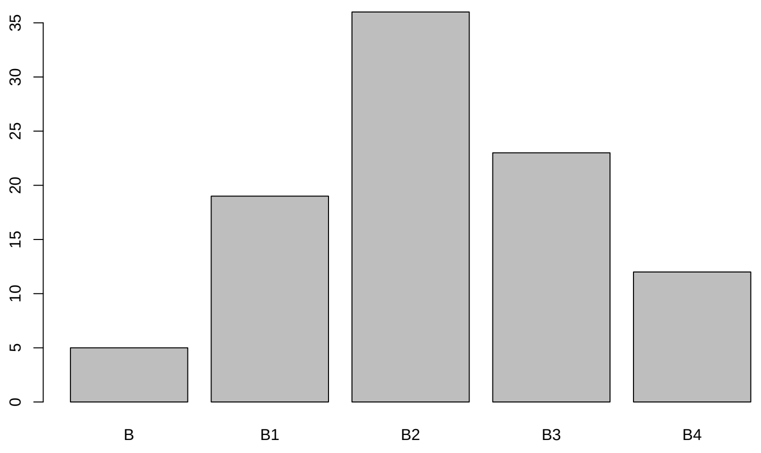

Figure 1.1. Class distribution of the stages of B-cell leukemia.

between the expression levels of each gene and the stage of the disease, our response variable, by means of a linear statistic based on the raw expression levels and a matrix of dummy-codings for the response. The statistics are standardized by their conditional expectation and variance [14] and the 100 genes associated with the largest standardized statistics are selected.

```
R> response <- BALL$BStage
R> I <- ncol(exprs(BALL))
R> expressions <- exprs(BALL)
R> Iindx <- 1:I
R> ymat <- model.matrix(~response - 1)
R> var_selection <- function(indx, p = 100) {
+     y <- ymat[indx, , drop = FALSE]
+     x <- expressions[, indx, drop = FALSE]
+     n <- nrow(y)
+     linstat <- x %*% y
+     Ey <- matrix(colMeans(y), nrow = 1)
+     Vy <- matrix(rowMeans((t(y) - as.vector(Ey))^2),
+         nrow = 1)
+     rSx <- matrix(rowSums(x), ncol = 1)
+     rSx2 <- matrix(rowSums(x^2), ncol = 1)
+     E <- rSx %*% Ey
+     V <- n/(n - 1) * kronecker(Vy, rSx2)
+     V <- V - 1/(n - 1) * kronecker(Vy, rSx^2)
+     stats <- abs(linstat - E)/sqrt(V)
+     stats <- do.call("pmax", as.data.frame(stats))
+     return(which(stats > sort(stats)[length(stats) -
+         p]))
+ }
R> selected <- var_selection(Iindx)
```

The function *var_selection* takes an index vector of observations between 1 and I and returns a vector of length `p` indicating which genes have been selected. Now, we are able to fit a random forest model to the data utilizing the package *MLInterfaces* which provides an unified interface to machine learning procedures including the *randomForest* package [15]. Here, we use the data of all observations except the Ith one as learning sample and obtain information from the model on the Ith observation.

```
R> pkgload <- require(MLInterfaces, quietly = TRUE)
R> rf <- randomForestB(BALL[selected, ], "BStage",
+     Iindx[-1], sampsize = I - 1)
R> print(rf)
```

```
MLOutput instance, method= randomForest
Call:
 randomForestB(exprObj = BALL[selected, ], classifLab = "BStage",
    trainInd = Iindx[-1], sampsize = I - 1)
predicted class distribution:
B2
 1
```

The framework for the evaluation of classifiers sketched in Section 1.5 can be implemented as follows. We loop over B bootstrap samples, perform a variable selection for the current bootstrap sample and fit four models to the selected genes of this bootstrap sample. First, a random forest model where only `mtry = 3` genes are evaluated in each node of the classification trees, bagging (which corresponds to random forest without random sampling of genes) and logitboost with $M = 100$ boosting iterations. In addition, the performance of a model that does not take any information about the expression levels into account is investigated. To be more specific, for each bootstrap sample we guess the class with maximal prior probability for all observations. The misclassification errors computed for each model and bootstrap sample are stored into a dataframe `performance`.

```
R> set.seed(290875)
R> B <- 100
R> if (!file.exists("ALLperformance.Rda")) {
+     performance <- as.data.frame(matrix(0, nrow = B,
+         ncol = 4))
+     colnames(performance) <- c("RF", "Bagg", "LBoost",
+         "Guess")
+     for (b in 1:B) {
+         bsample <- sample(Iindx, I, replace = TRUE)
+         selected <- var_selection(bsample)
+         rf3 <- randomForestB(BALL[selected, ],
+             "BStage", bsample, mtry = 3, sampsize = I)
+         predicted3 <- factor(rf3@predLabels, levels = levels(response))
+         performance[b, 1] <- mean(response[-bsample] !=
+             predicted3)
+         rfBagg <- randomForestB(BALL[selected,
+             ], "BStage", bsample, mtry = length(selected),
+             sampsize = I)
+         predictedBagg <- factor(rfBagg@predLabels,
+             levels = levels(response))
+         performance[b, 2] <- mean(response[-bsample] !=
+             predictedBagg)
+         lb <- logitboostB(BALL[selected, ], "BStage",
+             bsample, 100)
+         predictedlb <- factor(lb@predLabels, levels = levels(response))
+         performance[b, 3] <- mean(response[-bsample] !=
+             predictedlb)
+         performance[b, 4] <- mean(response[-bsample] !=
+             levels(response)[which.max(tabulate(response[bsample]))])
+     }
+     save(performance, file = "ALLperformance.Rda")
+ } else {
+     load("ALLperformance.Rda")
+ }
```

The distributions of the misclassification errors of all four models can be analyzed by the appropriate procedures for paired observations. The global null-hypothesis of equality of all three models can be tested using

```
R> friedman.test(as.matrix(performance))

        Friedman rank sum test

data:  as.matrix(performance)
Friedman chi-squared = 183.6596, df = 3, p-value <
2.2e-16
```

which leads to a rejection indicating that there are global differences between the performances of the four candidate models. This result allows us to conclude that there is a relationship between expression levels and the stage of B-cell leukemia. A parallel coordinate plot and boxplots help us to investigate where the differences come from (Figure 1.2).

```
R> layout(matrix(c(1, 2), ncol = 2))
R> matplot(1:ncol(performance), t(performance), type = "l",
+      col = 1, lty = 1, xlab = "", ylab = "Misclassification error",
+      axes = FALSE)
R> axis(1, at = 1:ncol(performance), labels = colnames(performance))
R> axis(2)
R> box()
R> boxplot(performance)
```

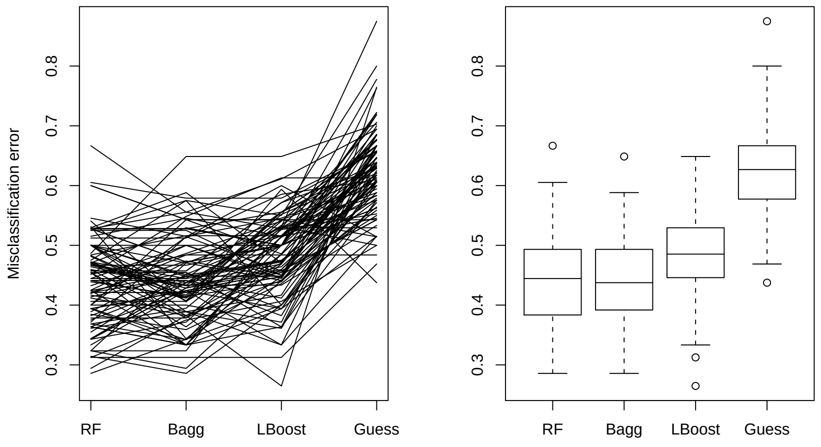

Figure 1.2. Parallel coordinates plot and boxplots of the misclassification errors of random forest (RF), bagging (Bagg), logitboost (LBoost) and guessing (Guess) for 100 bootstrap samples for the ALL data.

The figures lead to the impression that the three ensemble methods perform better than guessing without using information about gene expression profiles. Random forest and bagging seem to perform equally good, the amount of the difference can be inspected by confidence intervals, for example via

```
R> t.test(performance$RF, performance$Bagg, paired = TRUE,
+      conf.int = TRUE)

        Paired t-test

data:  performance$RF and performance$Bagg
t = -0.4623, df = 99, p-value = 0.6449
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.01627441  0.01012377
```

```
sample estimates:
mean of the differences
            -0.00307532
```

We emphasize again, see end of Section 1.5, that the Friedman test and the graphical illustrations indicate whether there are differences among the theoretical bootstrap distributions (with $B = \infty$), although we compute with $B = 100$ only.

1.6.2 Renal Cell Cancer

In this second application, we focus on the relationship between gene expression levels and response variables describing either clinical subtypes of renal cell cancer or, most interesting in a clinical setting, the survival time of the patients. The package *kidpack* offers data of a study by Sültmann et al. [16] explained in detail in Chapter XXX . The analysis is very similar to the steps described for the ALL data in Section 1.6.1. Chapter ref?

First, we load the package *kidpack* which offers the raw data in form of an *exprSet*.

```
R> set.seed(290875)
R> pkgload <- require(kidpack, quietly = TRUE)
R> data(eset)
R> pData(phenoData(eset))$type <- as.factor(eset$type)
```

The class distribution of clear cell renal cancer `ccRCC`, papillary renal cell cancer `pRCC` and chromophobe renal cell cancer `chRCC` is depicted in Figure 1.3. Again, a standardized linear statistic for each gene is applied to perform variable selection:

```
R> response <- eset$type
R> expressions <- exprs(eset)
R> I <- ncol(exprs(eset))
R> Iindx <- 1:I
R> ymat <- model.matrix(~response - 1)
R> var_selection <- function(indx, p = 100) {
+     y <- ymat[indx, , drop = FALSE]
+     x <- expressions[, indx, drop = FALSE]
+     n <- nrow(y)
+     linstat <- x %*% y
+     Ey <- matrix(colMeans(y), nrow = 1)
+     Vy <- matrix(rowMeans((t(y) - as.vector(Ey))^2),
+         nrow = 1)
+     rSx <- matrix(rowSums(x), ncol = 1)
+     rSx2 <- matrix(rowSums(x^2), ncol = 1)
+     E <- rSx %*% Ey
+     V <- n/(n - 1) * kronecker(Vy, rSx2)
```

```
R> barplot(table(pData(phenoData(eset))$type))
```

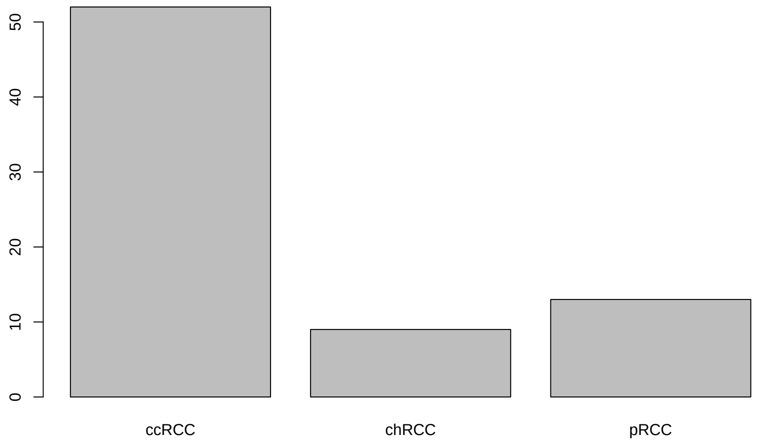

Figure 1.3. Class distribution of the renal cell cancer data

```
+     V <- V - 1/(n - 1) * kronecker(Vy, rSx^2)
+     stats <- abs(linstat - E)/sqrt(V)
+     stats <- do.call("pmax", as.data.frame(stats))
+     return(which(stats > sort(stats)[length(stats) -
+         p]))
+ }
R> selected <- var_selection(Iindx)
```

and a random forest model for the Ith patient can be fitted to 100 selected genes using

```
R> rf <- randomForestB(eset[selected, ], "type",
+     Iindx[-1], sampsize = I - 1)
R> rf

MLOutput instance, method= randomForest
Call:
 randomForestB(exprObj = eset[selected, ], classifLab = "type",
    trainInd = Iindx[-1], sampsize = I - 1)
predicted class distribution:
ccRCC
    1
```

The misclassification error for the four models (random forest, bagging, logitboost and guessing) applied to bootstrap samples of the data is computed along the following lines:

```
R> B <- 100
R> if (!file.exists("kidpackperformance.Rda")) {
+     performance <- as.data.frame(matrix(0, nrow = B,
+         ncol = 4))
+     colnames(performance) <- c("RF", "Bagg", "LBoost",
+         "Guess")
+     for (b in 1:B) {
+         bsample <- sample(Iindx, I, replace = TRUE)
+         selected <- var_selection(bsample)
+         rf3 <- randomForestB(eset[selected, ],
+             "type", bsample, mtry = 3, sampsize = I)
+         predicted3 <- factor(rf3@predLabels, levels = levels(response))
+         performance[b, 1] <- mean(response[-bsample] !=
+             predicted3)
+         rfBagg <- randomForestB(eset[selected,
+             ], "type", bsample, mtry = length(selected),
+             sampsize = I)
+         predictedBagg <- factor(rfBagg@predLabels,
+             levels = levels(response))
+         performance[b, 2] <- mean(response[-bsample] !=
+             predictedBagg)
+         lb <- logitboostB(eset[selected, ], "type",
+             bsample, 100)
+         predictedlb <- factor(lb@predLabels, levels = levels(response))
+         performance[b, 3] <- mean(response[-bsample] !=
+             predictedlb)
+         performance[b, 4] <- mean(response[-bsample] !=
+             levels(response)[which.max(tabulate(response[bsample]))])
+     }
+     save(performance, file = "kidpackperformance.Rda")
+ } else {
+     load("kidpackperformance.Rda")
+ }
```

Again, the global null-hypothesis of equality of the performance of the candidate models can be rejected

```
R> friedman.test(as.matrix(performance))

        Friedman rank sum test

data:  as.matrix(performance)
```

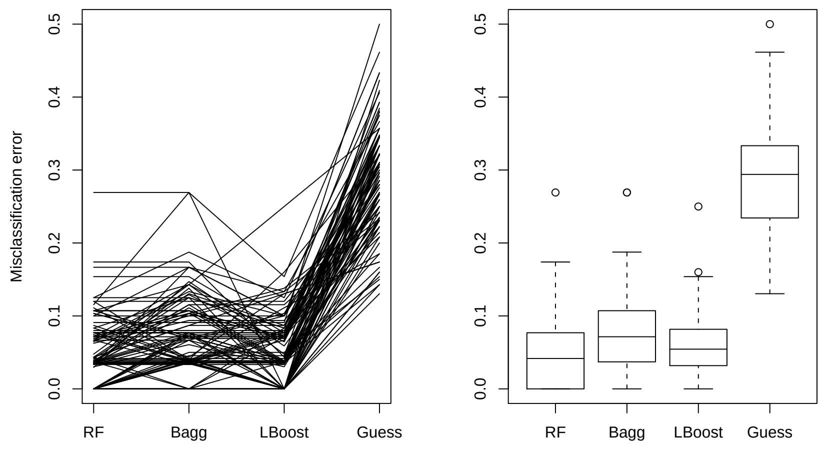

Figure 1.4. Parallel coordinates plot and boxplots of the misclassification errors of random forest (RF), bagging (Bagg), logitboost (LBoost) and guessing (Guess) for 100 bootstrap samples for the renal cell cancer data.

```
Friedman chi-squared = 217.7669, df = 3, p-value <
2.2e-16
```

which, in the light of Figure 1.4, can be explained by the superior performance of the ensemble methods compared to the model where we guess the prediction from the prior distribution of the classes itself. Random forests seem to perform a little bit better compared to bagging and show a performance similar to logitboost, as the confidence interval for the difference indicate:

```
R> t.test(performance$RF, performance$Bagg, paired = TRUE,
+      conf.int = TRUE)

        Paired t-test

data:  performance$RF and performance$Bagg
t = -5.3201, df = 99, p-value = 6.455e-07
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.03182856 -0.01453612
sample estimates:
mean of the differences
            -0.02318234
```

```
R> t.test(performance$RF, performance$LBoost, paired = TRUE,
+     conf.int = TRUE)

        Paired t-test

data:  performance$RF and performance$LBoost
t = -0.7676, df = 99, p-value = 0.4445
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.012479551  0.005517086
sample estimates:
mean of the differences
           -0.003481233
```

The same comment as at the end of Section 1.6.1 applies here as well.

1.7 Applications: Survival Analysis

Mainly in clinical studies, the disease free survival time or overall survival time is of major interest. For each observation, we are provided with the time for which each patient was under risk and whether a target event (recurrence, death) occurred or if the observation was stopped for other reasons (for example a lethal accident). More formally, the response variable is now bivariate with $y_i \in \mathbb{R}^+ \times \{0, 1\}$. We are interested in an assessment of the survival time to be expected for a patient with expression levels x, i.e., an assessment of the conditional probability that this patient will survive time t given the patients gene expression levels.

Basically, the ingredients of the bagging algorithm for the two-class problem can be applied to those problems as well. However, it is a challenge to aggregate the predictions of the multiple trees in order to come up with a ensemble prediction. A simple average of the predicted survival time did not prove to be of much use [17]. As an alternative, Hothorn et al. [18] suggested to aggregate observations instead of predictions directly and compute one single prediction based on aggregated observations. For each of the M survival trees fitted to bootstrap samples $\mathcal{L}_m, m = 1, \dots, M$, we extract the observations from the bootstrap sample which are element of the same terminal node the predictor value x of interest. Those observations, of course containing many tied values, are collected over all M bootstrap rounds, and then one single Kaplan-Meier curve is computed which serves as the ensemble's prediction.

```
R> pkgload <- require(survival, quietly = TRUE)
R> pkgload <- require(ipred, quietly = TRUE)
R> pkgload <- require(exactRankTests, quietly = TRUE)
R> remove <- is.na(pData(phenoData(eset))$survival.time)
```

```
R> seset <- eset[, !remove]
R> response <- Surv(seset$survival.time, seset$died)
R> response[response[, 1] == 0] <- 1
R> expressions <- exprs(seset)
R> exprDF <- as.data.frame(t(expressions))
R> I <- nrow(exprDF)
R> Iindx <- 1:I
```

The survival time of patients with renal cell cancer is now treated as the response variable of interest. First, we remove all observations where the survival time is missing (14 observations) from the data set and redefine zero survival times (one observation).

The response variable is now an object of class *Surv*. A slightly different test statistic needs to be used for the variable selection. Here, we use logrank scores as implemented in the *cscores* method from package *exactRankTests* and the expression values for each gene and apply a test statistic measuring the correlation between each gene and the logrank scores which is standardized by the conditional expectation and variance. Because of the small number of patients with information on the survival time being available, we only select 25 genes at a time.

```
R> var_selection <- function(indx, p = 25) {
+     y <- matrix(cscores(response[indx]), ncol = 1)
+     x <- expressions[, indx, drop = FALSE]
+     n <- nrow(y)
+     linstat <- x %*% y
+     Ey <- matrix(colMeans(y), nrow = 1)
+     Vy <- matrix(rowMeans((t(y) - as.vector(Ey))^2),
+         nrow = 1)
+     rSx <- matrix(rowSums(x), ncol = 1)
+     rSx2 <- matrix(rowSums(x^2), ncol = 1)
+     E <- rSx %*% Ey
+     V <- n/(n - 1) * kronecker(Vy, rSx2)
+     V <- V - 1/(n - 1) * kronecker(Vy, rSx^2)
+     stats <- abs(linstat - E)/sqrt(V)
+     stats <- do.call("pmax", as.data.frame(stats))
+     return(which(stats > sort(stats)[length(stats) -
+         p]))
+ }
R> selected <- var_selection(Iindx)
```

The ensemble of survial trees is fitted to the data using the *bagging* method (package *ipred* [19]). The predictions for each observation in the learning sample are computed based on trees whose corresponding bootstrap samples did not contain the specific observation (out-of-bootstrap prediction) and are therefore not affected by overfitting problems. In addition, we compute a simple Kaplan-Meier curve for the survival times via *survfit*. The 'model fit' of both procedures may be assessed by means of the Brier score for censored data [20] implemented in the function *sbrier*.

```
R> bagg <- bagging(response ~ ., data = exprDF[,
+     selected], ntrees = 100)
R> prKM <- predict(bagg)
R> sbrier(response, prKM)

integrated Brier score
             0.1284143
attr(,"time")
[1]  1 65

R> sbrier(response, survfit(response))

integrated Brier score
             0.2069643
attr(,"time")
[1]  1 65
```

This result seems to indicate that the survival ensemble fits the data better than a model without knowledge of the gene expression data and hence that there is information about the survival time covered by the expression levels. The predicted survival curves for each patient and the simple Kaplan-Meier estimate of all survival times are depicted in Figure 1.5 showing a reasonale differentiation between the patients. If this finding is reliable needs to be addressed by a benchmark comparison with a model that predicts the survival time without knowledge of the gene expression values, i.e., a simple Kaplan-Meier estimate.

```
R> set.seed(290875)
R> B <- 100
R> if (!file.exists("Survperformance.Rda")) {
+     performance <- as.data.frame(matrix(0, nrow = B,
+         ncol = 2))
+     colnames(performance) <- c("Bagging", "Kaplan-Meier")
+     for (b in 1:B) {
+         bsample <- sample(Iindx, I, replace = TRUE)
+         selected <- var_selection(bsample)
+         bagg <- bagging(response ~ ., data = exprDF[,
+             selected], subset = bsample, ntrees = 100)
+         pr <- predict(bagg, newdata = exprDF[-bsample,
+             ])
+         KM <- survfit(response[bsample])
+         performance[b, 1] <- sbrier(response[-bsample],
+             pr)
+         performance[b, 2] <- sbrier(response[-bsample],
R> plot(survfit(response), lwd = 4, conf.int = FALSE,
+     xlab = "Survival time in month", ylab = "Probability")
R> KMlines <- lapply(prKM, lines, lty = 2)
```

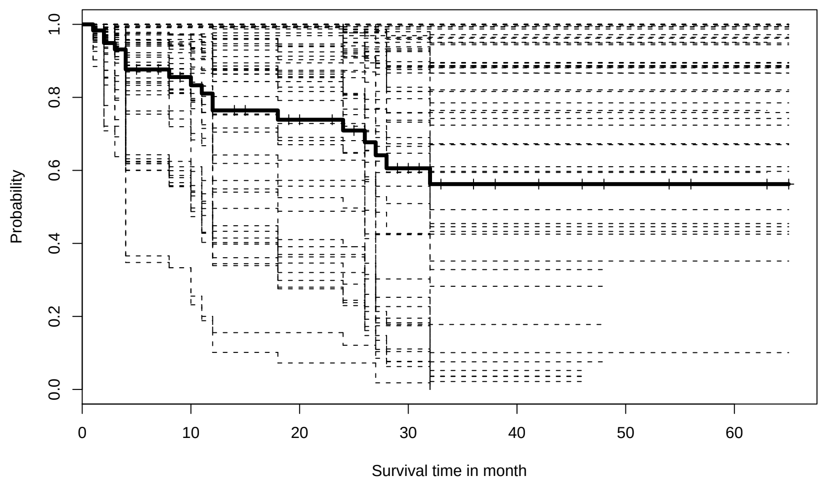

Figure 1.5. Out-of-bootstrap predicted survival curves for each patient (dashed lines) and overall Kaplan-Meier curve (thick solid line).

```
+              KM)
+     }
+     save(performance, file = "Survperformance.Rda")
+ } else {
+     load("Survperformance.Rda")
+ }
```

The visualizations of the Brier scores in Figure 1.6 indicate differences with respect to the variability but not with respect to the mean Brier score and hence we cannot conclude that the survival time of a patient can adequately modeled based on information derived from gene expression profiling. Note that this benchmark experiment is based on a learning sample of 60 patients, 42 of which are censored. This implies that, on average, only 38 unique patients are included in one bootstrap learning sample, which explains the large variability.

The same comment as at the end of Section 1.6.1 applies here as well.

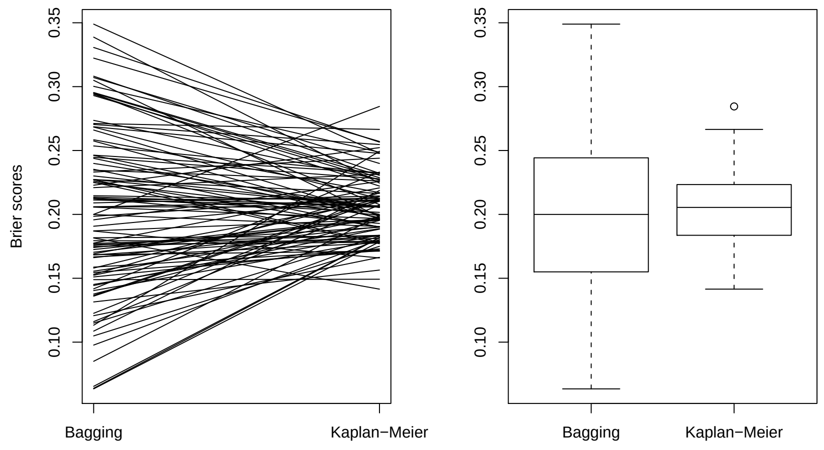

Figure 1.6. Parallel coordinates plot and boxplots of the Brier scores of 100 bootstrap samples for renal cell cancer survival.

1.8 Conclusions

Modeling the relationship between a clinically interesting response variable, such as tumor subtype of survival time, and gene expression levels of thousands of genes for only a small number of patients is a challenge to statistical methodology. The analyses in this chapter give an overview on ensemble methods for modeling high-dimensional gene expression data and illustrate both advantages and shortcomings. For typical sample sizes in the order of 100 patients, ensemble methods seem appropriate for constructing models for the prediction of tumor subtypes of renal cell cancer or stages of B-cell leukemia. It should be noted though that ensemble methods as used here are not more than black-box prediction methods. Modeling censored time-to-event data in such a high-dimensional setting is even more difficult, especially when a substantial number of patients are censored. Finally, we would like to mention that much research is currently conducted in the field of ensemble methods in both statistics and machine learning and one can expect further methodological developments and new software packages offering more functionality in the near future.

References

[1] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J.: Classification and regression trees.California, Wadsworth, 1984.

[2] Therneau, T. M. and Atkinson, E. J.An introduction to recursive partitioning using the rpart routine.Technical Report 61, Section of Biostatistics, Mayo Clinic, Rochester, 1997.

[3] Breiman, L.: Bagging predictors.Machine Learning.24(2): 123–140, 1996.

[4] Bühlmann, P. and Yu, B.: Analyzing bagging.The Annals of Statistics.30(4): 927–961, 2002.

[5] Breiman, L.: Random forests.Machine Learning.45(1): 5–32, 2001.

[6] Freund, Y. and Schapire, R. E.: Experiments with a new boosting algorithm.In Saitta, L. (Ed.): Machine Learning: Proceedings of the Thirteenth International Conference.San Francisco, Morgan Kaufmann, 1996, pp 148–156.

[7] Friedman, J., Hastie, T., and Tibshirani, R.: Additive logistic regression: A statistical view of boosting.The Annals of Statistics.28(2): 337–407, 2000.with Discussion.

[8] Dettling, M. and Bühlmann, P.: Boosting for tumor classification with gene expression data.Bioinformatics.19(9): 1061–1069, 2003.

[9] Dettling, M. and Bühlmann, P.: Finding predictive gene groups from microarray data.Journal of Multivariate Analysis.90(1): 106–131, 2004.

[10] Hastie, T., Tibshirani, R., and Friedman, J. H.: The Elements of Statistical Learning : Data Mining, Inference, and Prediction.Springer Verlag, 2001.

[11] Breiman, L.Out-of-bag estimation.Technical report, Statistics Department, University of California Berkeley, Berkeley CA 94708, 1996.

[12] Hothorn, T., Leisch, F., Zeileis, A., and Hornik, K.The design and analysis of benchmark experiments.Technical Report 82, SFB Adaptive Informations Systems and Management in Economics and Management Science, 2004.

[13] Chiaretti, S., Li, X., Gentleman, R., Vitale, A., Vignetti, M., Mandelli, F., Ritz, J., and Foa, R.: Gene expression profile of adult T-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival.Blood.103(7): 2771–2778, 2004.

[14] Strasser, H. and Weber, C.: On the asymptotic theory of permutation statistics.Mathematical Methods of Statistics.8: 220–250, 1999.

[15] Liaw, A. and Wiener, M.: Classification and regression by randomForest.R News.2(3): 18–22, December 2002.

[16] Sültmann, H., von Heydebreck, A., Huber, W., Kuner, R., Buneß, A., Vogt, M., Gunawan, B., Vingron, M., Füzesí, L., and Poustka, A.: Gene expression in kidney cancer is associated with novel tumor subtypes, cytogenetic abnormalities and metastasis formation.Clinical Cancer Research.2004.accepted.

[17] Dannegger, F.: Tree stability diagnostics and some remedies for instability.Statistics in Medicine.19(4): 475–491, 2000.

[18] Hothorn, T., Lausen, B., Benner, A., and Radespiel-Tröger, M.: Bagging survival trees.Statistics in Medicine.23(1): 77–91, 2004.

[19] Peters, A., Hothorn, T., and Lausen, B.: ipred: Improved predictors.R News.2(2): 33–36, June 2002.ISSN 1609-3631.

[20] Graf, E., Schmoor, C., Sauerbrei, W., and Schumacher, M.: Assessment and comparison of prognostic classification schemes for survival data.Statistics in Medicine.18(17-18): 2529–2545, November 1999.