Chapter 8

Classification Methods

In medical settings, groups of patients are often diagnosed into classes corresponding to types and/or subtypes of diseases. In bioinformatics, an important question is whether or not the diagnosis of a patient can be predicted by gene expression. A related question is which of the thousands of genes play an important role in the prediction of class membership. These important genes are often called *biomarkers*. Other potential *biomarkers* include microRNA expression and protein concentrations. In general, it's possible to attempt to associate any combination of cell states or cellular quantitative measurements to a phenotypic class.

Many classification methods have been developed for various scientific purposes. In bioinformatics, methods such as *recursive partitioning tree*, *support vector machine*, and the *neural network* are frequently applied to solve classification problems.

In this chapter, we will describe what recursive partitioning is and how to use it. Two other methods to predict disease class from gene expression data are the *support vector machine (SVM)* and the *neural network (NN)*. We will briefly explain what these methods are about and how they can be applied to gene expression data. For all the classification methods, the same training and testing sets will be used to evaluate their predictive accuracies. To evaluate the quality of any predictions, the fundamental concepts of *sensitivity* and *specificity* are frequently used. The specificity can be summarized in a single number by the *area under the curve (AUC)* of a *receiver operating characteristic (ROC)* curve - which will be explained and illustrated.

8.1 Classification of microRNA

The subject of making a correct medical diagnosis is highly similar to that of correctly classifying microRNA.

Example 1: Classification of microRNA. MicroRNA are small RNA molecules with important functions in cell growth and disease development. In order to identify microRNAs from arbitrary RNA sequences, their characterizing properties are used to distinguish them from non-microRNA molecules. One of these properties is that microRNAs have the capacity to fold in a certain hairpin type of structure. Such a structure typically exhibits a small minimum folding energy (Zuker, 2003; Zuker & Stiegler, 1981). This property can be used as a test to discriminate microRNAs from non-microRNAs (Bonnet, et al., 2004). Given a set of 3424 different microRNAs, the minimum folding energy was computed for each of these. Next, for each microRNA the order of the nucleotides was shuffled with replacement 1000 times. This yielded 1000 differently shuffled sequences of nucleotides per microRNA for which the minimum folding energy is computed.[1] Per microRNA, the 1001 energy values were arranged in increasing order, similar to the empirical distributions in the previous chapter. Then the number of minimum folding energies below that of the original microRNA is counted and divided by 1001 to obtain a p-value. If the minimum folding energy of the original microRNA is the smallest, then the empirical p-value is zero. This procedure yielded a total of 3424 p-values. The number of sequences with p-values below the threshold value of 0.01 is given in Table 8.1. The same procedure is conducted for non-microRNA molecules which were chosen to have similar length and nucleotide percentages as the microRNAs.

Table 8.1: Frequencies empirical p-values lower than or equal to 0.01.

	test positive $p \leq 0.01$	test negative $p > 0.01$	total
microRNA	2973	451	3424
non-microRNA	33	3391	3424
total	3006	3842	6848

[1] I am obliged to Sven Warris for computing the minimum energy values.

From the frequency Table 8.1, the *sensitivity*, *specificity*, and *predictive power* can be computed in order to evaluate the quality of the test. The sensitivity is the probability that the test is positive given that the sequence is a microRNA (true positive) Thus,

$$\texttt{Sensitivity} = P(\text{true positive}) = P(\text{test positive}|\text{microRNA}) = \frac{2973}{3424} = 0.8682.$$

The specificity is the probability that the test is negative given that the sequence is not a microRNA (true negative). Thus,

$$\texttt{Specificity} = P(\text{true negative}) = P(\text{test negative}|\text{non-microRNA}) = \frac{3391}{3424} = 0.9903.$$

For practical applications of a test, the predictive power is of crucial importance. In particular, the *predictive value positive* is the probability that the sequence is a microRNA given that the test is positive. That is,

$$\texttt{Predictive value positive} = PV^{+} = P(\text{microRNA}|\text{test positive}) = \frac{2973}{3006} = 0.9890$$

Thus, when the test is positive we are 98.90% certain that the sequence is indeed a microRNA. The *predictive value negative* is the probability that the sequence is not a microRNA given that the test is negative:

$$\texttt{Predictive value negative} = PV^{-} = P(\text{non-microRNA}|\text{test negative}) = \frac{3391}{3842} = 0.8826.$$

Thus, when the test is negative we are 88.26% certain that the sequence is not a microRNA. From the estimated conditional probabilities it can be concluded that the test performs quite well in discriminating microRNAs from non-microRNAs.

8.2 ROC curves

In Chapter 2, we have observed that with the Golub et al. (1999) data the expression values of the gene CCND3 (Cyclin D3) tend to be greater for ALL patients. Therefore, we may use CCND3 expression data as a test for predicting ALL using a certain CCND3 cutoff value or threshold for classification. In particular, for gene expression values larger than a certain cutoff

we declare the test "positive" in the sense of indicating ALL. Note that the corresponding true and false positives can be computed for each possible cutoff value - which is how a ROC curve is calculated. To briefly indicate the origin of the terminology, imagine that the test results for each cutoff value are a characteristic received by an operator. The receiver operating characteristic (ROC) is a curve where the false positive rates are depicted horizontally and the true positive rates vertically for all the possible cutoff values. Since the ROC curve is independent of any one threshold value, it is considered to be a non-parametric measurement of predictive power. The larger the area under the ROC curve (AUROC), the better the test is since then low false positive rates combine together with large true positive rates.[2] These ideas are illustrated by several examples below.

Example 1: Classification via CCND3 expression. For the sake of illustration, we consider the prediction of ALL from the expression values for the gene CCND3 (Cyclin D3) from Golub et al. (1999). Let's consider the cutoff point 1.27. For this cutoff point we can produce a table with TRUE/FALSE frequencies for predicting ALL/not ALL:

```
> data(golub, package = "multtest")
> golubLabels <- factor(golub.cl,levels=0:1,labels= c(" ALL","not ALL"))
> ccnd3 <- grep("CCND3",golub.gnames[,2], ignore.case = TRUE)
> golubPredictor <- factor(golub[ccnd3,]>1.27,levels=c("TRUE","FALSE"),
   labels=c("ALL","notALL"))
> table(golubPredictor,golubLabels)
            golubLabels
golubPredictor  ALL not ALL
  ALL      25       1
  notALL    2      10
```

There are 25 ALL patients with expression values greater than or equal to 1.27, and thus the true positive rate is $25/27 = 0.93$. For this cutoff value there is one false positive because one patient without ALL has a score larger than 1.27. Hence, the false positive rate is $1/11 = 0.09$.

Example 2: Constructing a ROC curve. First, the expression values for gene CCND3 (Cyclin D3) from the Golub et al. (1999) data are sorted in decreasing order, see Table 8.2. The procedure to draw the ROC curve starts with the cutoff point set to infinity. Obviously, there are no expression values equal to infinity, so no patient tests positive. Next, the cut off point

[2] More detailed information can be obtained from a wikipedia search using "ROC curve".

Table 8.2: Ordered expression values of gene CCND3 (Cyclin D3), index 2 indicates ALL, 1 indicates AML, cutoff points, number of false positives, false positive rate, number of true positives, true positive rate.

	data	index	cutoff	fp	fpr	tp	tpr
1			Inf	0	0.00	0	0.00
2	2.77	2	2.77	0	0.00	1	0.04
3	2.59	2	2.59	0	0.00	2	0.07
4	2.45	2	2.45	0	0.00	3	0.11
5	2.37	2	2.37	0	0.00	4	0.15
6	2.34	2	2.34	0	0.00	5	0.19
7	2.31	2	2.31	0	0.00	6	0.22
8	2.18	2	2.18	0	0.00	7	0.26
9	2.18	2	2.18	0	0.00	8	0.30
			⋮				
31	1.02	1	0.89	4	0.36	26	0.96
32	0.89	1	0.83	5	0.45	26	0.96
33	0.83	1	0.74	6	0.55	26	0.96
34	0.74	1	0.64	7	0.64	26	0.96
35	0.64	1	0.49	8	0.73	26	0.96
36	0.49	1	0.46	8	0.73	27	1.00
37	0.43	1	0.43	9	0.82	27	1.00
38	0.13	1	0.13	10	0.91	27	1.00
39	−0.74	1	−0.74	11	1.00	27	1.00

2.77 is chosen and values greater than or equal to 2.77 now test as positive. This yields one true positive implying a true positive rate of $1/27$ - see the second row of Table 8.2. For this cutoff value there are no negatives, and thus the false positive rate is zero.

Now consider cutoff point 1.52. There are 22 ALL patients with expression values greater than or equal to 1.52, and thus the true positive rate is $22/27 = 0.81$. For this cutoff value there are no false positives because all patients without ALL have expression values lower than 1.51. Hence, the false positive rate is 0 and the true positive rate is 0.81. To indicate this, there is a vertical line drawn in the ROC curve from point $(0, 0)$ to point $(0, 0.81)$ in Figure 8.1.

Now consider the next cutoff point 1.45. There are 22 ALL patients with expression values greater than or equal to 1.45, and thus the true positive rate is again $22/27$=0.81. However, there is one patient without ALL having

expression value 1.45, who therefore receives a positive test. Hence, the number of false positives increases from zero to one, which implies a false positive rate of $1/11 = 0.09$. In the ROC curve this is indicated by the point $(0.09, 0.81)$ and the horizontal line from $(0, 0.81)$ to $(0.09, 0.81)$ in Figure 8.1.

This process goes on (see Table 8.2) until the smallest data point -0.74 is chosen as the cutoff point. For this cutoff point all patients test positive, and thus the false positive rate is $11/11$ and the true positive rate is $27/27$. This is indicated by the end point $(1, 1)$ in the plot at the top on the right hand side.

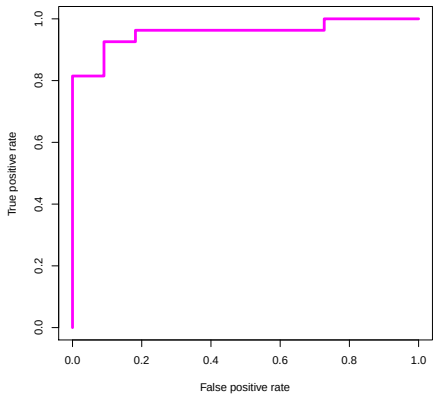

Figure 8.1: ROC curve for expression values of CCND3 (Cyclin D3).

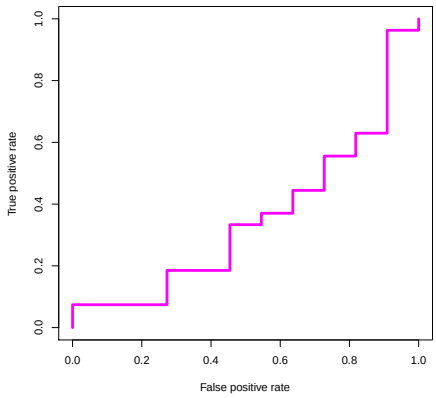

Figure 8.2: ROC curve for expression values of gene Gdf5.

It is obviously helpful to use a bioinformatics package for producing an ROC curve such as in Figure 8.1. To do so we construct an appropriate factor with the value TRUE for ALL and FALSE for not ALL `golubLabels` and use functions from the `ROCR` package:

```
> library(ROCR)
> golubLabels <- factor(golub.cl,levels=0:1,labels= c("TRUE","FALSE"))
> ccnd3 <- grep("CCND3",golub.gnames[,2], ignore.case = TRUE)
> pred <- prediction(golub[ccnd3,], golubLabels)
> perf <- performance(pred, "tpr", "fpr" )
> plot(perf,
+      lwd=4,
+      col="magenta")
> index <- c(1,1)
> gdf5 = grep("GDF5",golub.gnames[ ,2], ignore.case = TRUE)
> gdf5
[1] 2058
```

```
> data <- c(sort(golub[gdf5,1:27],decreasing = TRUE),sort(golub[gdf5,28:38],decreasing =
    ↪ TRUE))
> for (i in 1:38) if (golub.cl[i]==0) index[i] <- 2 else index[i] <-golub.cl[i]
> pred <- prediction(data, index)
> perf <- performance( pred, "tpr", "fpr" )
> plot(perf,
+      lwd=4,
+      col="magenta")
>
> index <- c(1,1)
> gdf5 = grep("GDF5",golub.gnames[ ,2], ignore.case = TRUE)
> gdf5
[1] 2058
> data <- c(sort(golub[gdf5,1:27],decreasing = TRUE),sort(golub[gdf5,28:38],decreasing =
    ↪ TRUE))
> for (i in 1:38) {
+     if (golub.cl[i] == 0) {
+         index[i] <- 2 }
+     else {
+         index[i] <-golub.cl[i]
+     }
+ }
> pred <- prediction(data, index)
> perf <- performance(pred, "tpr", "fpr" )
> plot(perf,
+      lwd=4,
+      col="magenta")
```

It seems clear that the gene expression values are better in testing for ALL when the ROC curve is very steep in the beginning and attains its maximum value of 1.0 quickly. In that case, the true positive rate is large with a corresponding small false positive rate - which is ideal. A method to express the predictive accuracy of a test in a single number is by the area under the curve (AUC). Using the function `performance(pred,"auc")`, we obtain that the area under the ROC curve is 0.96, which is very close to the optimum max of 1.0. Hence, the expression values of CCND3 (Cyclin D3) are very suitable for discrimination between ALL and not ALL (AML).

For contrast, the ROC curve for the expression values of gene Gdf5 is also calculated. In Figure 8.2, we can see that the true positive rate for Gdf5 expression is much lower as one moves on the horizontal axis from left to right. This corresponds to the area under the curve of 0.35, which is far less than 0.96. In fact, this Gdf5-expression classifier is performing worse than random guessing - which corresponds to an AUC of 0.5. We can improve the Gdf5-expression classifier significantly by reversing the classification labels. With the ability to just reverse the classification labels, each classifier has an AUC lower bound of 0.5 as well as the AUC upper cound of 1.0. This illustrates that genes may exhibit large differences with respect to prediction of the

disease status of patients. For example, CCND3 expression is a biomarker for ALL in the Golub data, but Gdf5 is not.

In practical applications, one is often interested in a single optimal cut-off value and in possibly combining several predictors in a decision scheme to increase accuracy. This is done using classification trees.

8.3 Classification trees

The purpose of classification is to allocate organelles, cells, tissues, organisms, etc. into classes on the basis of measurements of attributes. For instance, in the case of the Golub et al. (1999) data, the organisms are 38 patients which have expression measurements for 3,051 genes. The classes consist of different diagnoses of patients into the ALL class (27 patients) and the AML class (11 patients). A tree model resembles that of a linear model, where the criterion is the factor indicating class membership and the predictor variables are the gene expression values. In the case of the Golub et al. (1999) data, the gene expression values $\{x_1, \cdots, x_{38}\}$ can serve as predictors to form a decision tree. An example decision is, if $x_j < t$ then the patient j is AML, and otherwise if $x_j \geq t$ then patient j is ALL. Obviously, the threshold value t on which the decision is based should be optimal given the predictor. The optimal value can be estimated by a regression tree (Breiman et al., 1984; Chambers & Hastie, 1992; Venables, & Ripley, 2000), which is implemented in the `rpart` package (Therneau & Atkinson, 1997).

A training set is used to estimate the threshold values that construct the tree. When many predictor variables are involved, 3051 in this instance, then we have a tremendous gene (variable) selection problem. Fortunately, the `rpart` package automatically selects genes which are important for classification and neglects others. Another common problem is that of *overfitting* when too many decision nodes are added to the tree to increase prediction accuracy. When such decision nodes are specific to a particular training sample set, they cannot be generalized to other samples and therefore have limited scientific value. The problem of overfitting to the training data is compounded when the sample size is small or when noise is present in the data. Prevention of such overfitting in a classification tree is called *pruning* and is automatically done by the `rpart()` function. We will illustrate how to use classification trees with some examples.

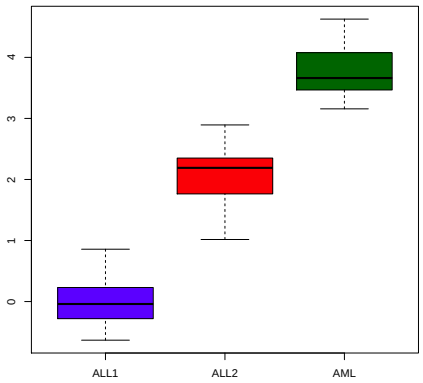

Figure 8.3: Boxplot of expression values of geneA for each leukemia class.

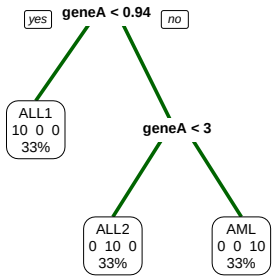

Figure 8.4: Classification tree using geneA for classifying three classes of leukemia.

Example 1: Optimal gene expression thresholds. Suppose microarray expression data are available with respect to patients suffering from three types of leukemia abbreviated as ALL1, ALL2, and AML. GeneA has expression values from the populations (patient groups) $N(0, 0.5^2)$ for ALL1, $N(2, 0.5^2)$ for ALL2, and $N(4, 0.5^2)$ for AML. The code below generates thirty expression values for geneA, the patients of the three disease classes, and the estimates of the classification tree:

```
> library(rpart)
> library(rpart.plot)
> set.seed(123); n<-10 ; sigma <- 0.5
> factor <- factor(c(rep(1,n),rep(2,n),rep(3,n)))
> levels(factor) <- c("ALL1","ALL2","AML")
> geneA <- c(rnorm(10,0,sigma),rnorm(10,2,sigma),rnorm(10,4,sigma))
> data <- data.frame(factor,geneA)
> rpartFit  <-  rpart(fac ~ geneA, method="class",data=data)
> boxplot(geneA ~ factor, col=c("blue", "red", "darkgreen"))
> prp(rpartFit,
+     branch.lwd=4, # wide branches
+     branch.col="darkgreen",
+     extra=101)    # plot label numbers and the percentage of observations
```

From the boxplot in Figure 8.3, we can see that there is no overlap of geneA expression between the three classes. This makes geneA an ideal predictor for separating the patients into the classes. By the construction of the gene expression values $x_1, \cdots, x_{30}$, we would expect the following classifica-

tion tree: (1) if $x_i < 1$ then ALL1, else (2) if x_i is in the interval $[1, 3]$ then ALL2, else (3) if $x_i > 3$ then AML. From the classification tree in Figure 8.4 we can see that the estimated splits are close to our expectations: (1) if $x_i < 0.971$ then ALL1, else (2) if x_i is in $[0.9371, 3.025]$ then ALL2, else (3) if $x_i > 3.025$ then AML. The tree consists of three leaves (nodes) and two splits. The predictions of the patients into the three classes perfectly match their true disease status.

Obviously, such an ideal single gene may not exist since expression values between the disease classes often overlap. In such cases, more genes may be used to build the classification tree and improve predictive accuracy.

Example 2: Gene selection. Another situation is where geneA discriminates between ALL and AML, geneB between ALL1 patients and ALL2 or AML patients, and geneC does not discriminate at all. To simulate this setting, we generate expression values for geneA from $N(0, 0.5^2)$ for both ALL1 and ALL2, and from $N(2, 0.5^2)$ for AML patients. Next, we generate expression values for geneB from $N(0, 0.5^2)$ for ALL1 and from $N(2, 0.5^2)$ for ALL2 and AML. Finally, we generate for geneC expression from $N(1, 0.5^2)$ for ALL1, ALL2, and AML. We generate these three simulated gene expressions and construct the corresponding classification tree below:

```
> set.seed(123)
> n<-10 ; sigma <- 0.5
> factor <- factor(c(rep(1,n),rep(2,n),rep(3,n)))
> levels(factor) <- c("ALL1","ALL2","AML")
> geneA <- c(rnorm(20,0,sigma),rnorm(10,2,sigma))
> geneB <- c(rnorm(10,0,sigma),rnorm(20,2,sigma))
> geneC <- c(rnorm(30,1,sigma))
> data <- data.frame(factor,geneA,geneB,geneC)
> rpartFit <- rpart(factor ~ geneA + geneB + geneC, method="class",data=data)
> boxplot(geneA ~ factor, col=c("blue", "blue", "red"))
> prp(rpartFit,
+     branch.lwd=4, # wide branches
+     branch.col="blue",
+     extra=101)    # plot label numbers and the percentage of observations
```

The addition in the model notation "factor geneA + geneB + geneC" in the `rpart()` function specifies a linear model where the dependent variable "factor" is linearly dependent on the expression of the three independent genes: geneA, geneB, and geneC. In addition, there are no interaction terms between the three genes. Thus, this model assumes that the three genes affect the factor independent of each other. It is convenient to collect the data in the form of a data.frame.

From the resultant boxplot in Figure 8.5, we can see that geneA discriminates well between ALL and AML, but not between ALL1 and ALL2. The expression values for geneB discriminate well between ALL1 and ALL2, whereas those of geneC do not discriminate at all. The latter can also be seen from the estimated tree in Figure 8.6, where geneC plays no role at all. In the decision treee, expression values of geneA larger than 1.025 are predicted as AML and smaller ones as ALL. Expression values of geneB smaller than 0.9074 are predicted as ALL1 and larger as ALL2. Hence, geneA separates well within the ALL class. This example illustrates that `rpart()` automatically selects the best genes (variables) which play a role in the classification.

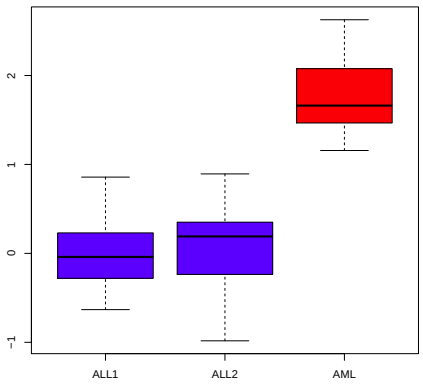

Figure 8.5: Boxplot of expression values of genes A for each leukemia class.

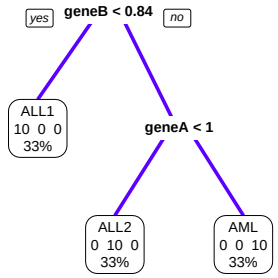

Figure 8.6: Classification tree using the expression values from genes A, B, and C for the classification of ALL1, ALL2, and AML patients.

Example 3: Classification by CCND3 gene expression. From the various visualizations and statistical testing in the previous chapters, we can conjecture that CCND3 (Cyclin D3) gene expression forms a suitable predictor for discriminating between ALL and AML patients. However, note from Figures 2.3 and 8.7 that there is some overlap between the expression values for the ALL and the AML patients. Therefore, a perfect classification is not possible. We use the function `rpart()` to perform the regression partitioning:

```
> library(rpart); library(multtest); data(golub)
> golubFactor <- factor(golub.cl,levels=0:1, labels= c("ALL","AML"))
> ccnd3 <- grep("CCND3",golub.gnames[,2], ignore.case = TRUE)
> golubRpart <- rpart(golubFactor ~ golub[ccnd3,] , method="class")
> predictedclass <- predict(golubRpart, type="class")
> table(predictedclass, golubFactor)
              golubFactor
predictedclass ALL AML
           ALL  25   1
           AML   2  10
```

Note that $(25 + 10)/38 \cdot 100\% = 92.10\%$ of the ALL/AML patients are correctly classified by CCND3 (Cyclin D3) gene expression. With the function `predict(golubRpart,type="class")`, the predictions from the regression tree for the patients into the two classes can be obtained. The factor `golubFactor` contains the levels `ALL` and `AML` corresponding to the diagnosis to be predicted. The predictor variable consists of the expression values of gene CCND3 (Cyclin D3). The output of recursive partitioning is assigned to an object called `golubRpart`, which is a list from which further information can be extracted by suitable functions. A summary can be obtained as follows:

```
> summary(golubRpart)
Call:
rpart(formula = golubFactor ~ golub[ccnd3, ], method = "class")
  n= 38

         CP nsplit rel error    xerror      xstd
1 0.7272727      0 1.0000000 1.0000000 0.2541521
2 0.0100000      1 0.2727273 0.5454545 0.2043460

Node number 1: 38 observations,    complexity param=0.7272727
  predicted class=ALL  expected loss=0.2894737
    class counts:    27    11
   probabilities: 0.711 0.289
  left son=2 (26 obs) right son=3 (12 obs)
  Primary splits:
      golub[ccnd3, ] < 1.198515 to the right, improve=10.37517, (0 missing)

Node number 2: 26 observations
  predicted class=ALL  expected loss=0.03846154
    class counts:    25     1
   probabilities: 0.962 0.038

Node number 3: 12 observations
  predicted class=AML  expected loss=0.1666667
    class counts:     2    10
   probabilities: 0.167 0.833

26
[1] 0.03846154
```

The expected loss in prediction accuracy of node number 2 is 1/26 and

that of node number 3 is 2/12. These percentages equal the probabilities from the class counts. Each primary split gives the estimated threshold value. To predict the class of the individual patients we can use the function `predict()`:

```
> predict(golubRpart,type="class")
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL AML ALL ALL ALL
 21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
AML ALL ALL ALL ALL ALL ALL AML ALL AML AML AML AML AML AML AML AML AML
Levels: ALL AML
```

Hence, patients 17 and 21 are erroneously predicted as AML and patient 29 is erroneously predicted to be in the ALL class. A more precise output is obtained by asking for the probability of class membership:

```
> predict(golubRpart, type="prob")
        ALL       AML
1 0.9615385 0.03846154
2 0.9615385 0.03846154
etc.
> boxplot(golub[1024,] ~ golubFactor, col=c("blue", "red"))
> prp(golubRpart,
+      branch.lwd=4, # wide, thick branches
+      branch.col="blue",
+      extra=101)    # plot label numbers and the percentage of observations
```

Therefore, the estimated probability of patient 21 to have ALL is 0.16 and to have AML is 0.83.

Example 4: Gene selection of the Golub (1999) data. Through recursive partitioning it is possible to select among the genes those which give the best classification predcitive rates. To do so, we have to specify the gene expressions as the variables (columns). For this we use the transposition operator `t()`. To facilitate reading of the output, we add `gene 1` to `gene 3051` as column names:

```
> library(rpart);data(golub); library(multtest)
> row.names(golub)<- paste("gene", 1:3051, sep = "")
> golubData <- data.frame(t(golub[1:3051,]))
> golubFactor <- factor(golub.cl,levels=0:1, labels= c("ALL","AML"))
> golubRpart  <-  rpart(golubFactor~., data=golubData, method="class", cp=0.001)
> prp(golubRpart,
+      branch.lwd=4, # wide, thick branches
+      branch.col="blue",
+      extra=101)    # plot label numbers and the percentage of observations
> golub.gnames[896,]
```

Inspection of the plot yields gene "FAH Fumarylacetoacetate" as the best predictor by which the two classes of patients can be predicted perfectly. Note that this is proven to be true only for this training set and that "FAH Fu-

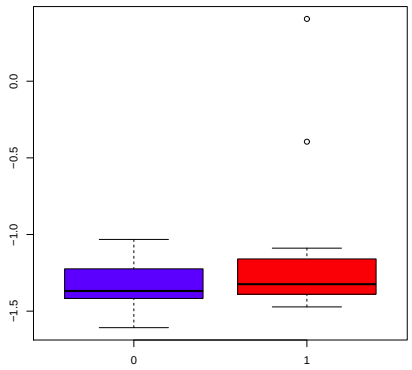

Figure 8.7: Boxplot of expression values of gene CCND3 (Cyclin D3) for ALL and AML patients

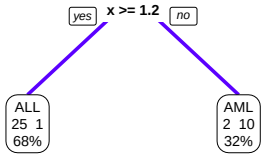

Figure 8.8: Classification tree of expression values from gene CCND3 (Cyclin D3) for classification of ALL and AML patients.

marylacetoacetate" may not hold as a good predictor for any new data.

Now, to further illustrate the possibilities of different classification methods we will use the ALL data collected by Chiaretti, et al. (2004).

Example 5: Application to the Chiaretti (2004) data. With the ALL data, we want to predict the diagnosis of B-cell state B1, B2, or B3 from the gene expressions. Since the complete set of 12,625 gene expressions is too large, we select the genes with different means over the patient groups. It is obvious that only these genes can contribute to the prediction of the disease states. In particular, we select the genes with an ANOVA p-value smaller than 0.000001:

```
> library("hgu95av2.db");library(ALL);data(ALL)
> ALLB123 <- ALL[,ALL$BT %in% c("B1","B2","B3")]
> anova.pValue <- apply(exprs(ALLB123), 1, function(x) anova(lm(x ~ ALLB123$BT))$Pr[1])
> names <- featureNames(ALL)[anova.pValue<0.000001]
> symb <- mget(names, env = hgu95av2SYMBOL)
> ALLBTnames <- ALLB123[names, ]
> probeData <- as.matrix(exprs(ALLBTnames))
> row.names(probeData)<-unlist(symb)
```

The probe symbols are extracted from the `hgu95av2SYMBOL` environment

and used as row names to facilitate readability in the resulting tree. Using the `anova()` p-values, we select 78 patients and 82 probes. The recursive partitioning to find the tree can be performed using the `rpart()` function:

```
> diagnosed <- factor(ALLBTnames$BT)
> rpartFit <- rpart(diagnosed ~ ., data = data.frame(t(probeData)))
> prp(rpartFit,
+     branch.lwd=4, # wide, thick branches
+     branch.col="blue",
+     extra=101)    # plot label numbers and the percentage of observations
> rpartPredict <- predict(rpartFit, type="class")
> table(rpartPredict,diagnosed)
           diagnosed
rpartPredict B1 B2 B3
       B1 17  2  0
       B2  1 33  5
       B3  1  1 18
```

The rows to the left in the table give the frequencies of the predicted B cell stages, and the column labels on top give the diagnosed B cell stages from the factor. Also, the matrix with the frequencies of the predicted and true patient status is often called a "confusion table". The resulting tree in Figure 8.9 should be read as follows: (1) if MME gene expression is strictly less than 8.395 then the patient is predicted to be in state (class) B1, else (2) if the expression of LSM6 less than 4.192 then the predicted state is B2, else (3) if it is larger then the predicted state it is B3.

The misclassification rate is $10/78 = 0.1282051$, which is low, but not zero. It may happen that the probability of the predicted class is close to that of the diagnosed. An overview of the latter can be obtained as follows:

```
> predicted.class <- predict(rpartFit, type="class")
> predicted.probabilities <- predict(rpartFit, type="prob")
> out <- data.frame(predicted.probabilities,predicted.class, diagnosis=factor(ALLBTnames$BT
    ↪ ))
> print(out,digits=2)
         B1   B2   B3 predicted.class diagnosis
01005 0.026 0.85 0.13              B2        B2
01010 0.026 0.85 0.13              B2        B2
04006 0.895 0.11 0.00              B1        B1
04007 0.026 0.85 0.13              B2        B2
04008 0.895 0.11 0.00              B1        B1
04010 0.050 0.05 0.90              B3        B1
04016 0.895 0.11 0.00              B1        B1
06002 0.026 0.85 0.13              B2        B2
08001 0.026 0.85 0.13              B2        B2
08011 0.026 0.85 0.13              B2        B3
08012 0.026 0.85 0.13              B2        B3
08018 0.050 0.05 0.90              B3        B3
08024 0.895 0.11 0.00              B1        B2
09008 0.026 0.85 0.13              B2        B3
...
```

For instance, the misclassified sixth patient has probability .90 of being in class B3 and probability .05 of being in class B1 - which is the diagnosed disease state.

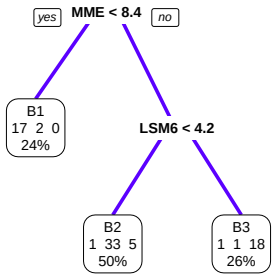

Figure 8.9: Classification tree generated by rpart() on the ALL B-cel 123 data.

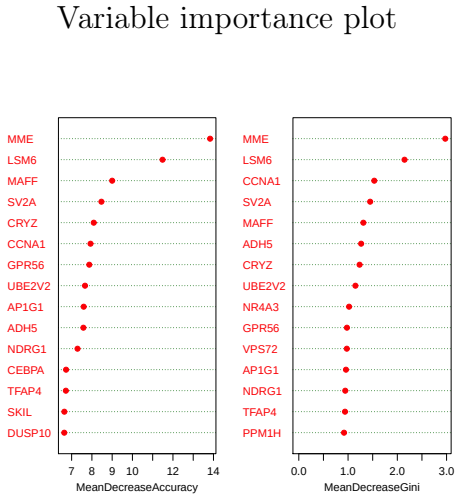

Figure 8.10: Dotchart of variable importance as measured by a Random Forest on the ALL B-cell 123 data.

Note the reduction in variables from 29 to 2 during the construction of the tree. During tree construction, typically only one gene is chosen from any given set of correlated gene expressions (variables). For example, when the first gene is selected for the first split, then any highly similar genes to the first will not be selected since they would not add any new predictive value to the tree. To help illustrate this point, we will leave out the variables selected from the data and redo the analysis.

A generally applied manner to evaluate a classification model is by its predictive accuracy with respect to a future data set. When such a future data set is not available, it is common practice to split the available data into two parts: A training set and a test (or validation) set. The model is estimated from the training set and then the model is used to predict the class of the patients in the test (validation) set. Then a confusion matrix is constructed with the frequencies of true classes against predicted classes. Next, the misclassification rate can be computed to evaluate the predictive accuracy. Generally, this is a method to detect for overfitting - where the

model estimates are too specific to the training set and therefore don't generalize well to future data sets.

Example 6: Separate training and test sets. With the B-cell ALL data with States 1, 2, and 3, we randomly split the patients into two separate sets. The 78 patients in State 1, 2 or 3 can be split in two halves as follows:

```
> training <- sample(1:78, 39, replace = FALSE)
> testing <- setdiff(1:78,training)
> df <- data.frame(t(probeData))
> rpartFit <- rpart(diagnosed ~ ., data = df, subset=training)
> rpart.predictor.t <- predict(rpartFit, df[training,], type="class")
> table(rpart.predictor.t,factor(ALLBTnames$BT[training]))
rpart.predictor.t B1 B2 B3
          B1 11  1  0
          B2  0 12  0
          B3  0  1 14
> rpart.predictor.v <- predict(rpartFit,df[testing,], type="class")
> table(rpart.predictor.v,factor(ALLBTnames$BT[testing]))
rpart.predictor.v B1 B2 B3
          B1  6  1  0
          B2  1 19  3
          B3  1  2  6
```

The misclassification rate in the training set is $2/39 = 0.05$ and in the test set is $8/39 = 0.21$. Note that the differences mainly occur between State 2 and 3. The predictive accuracy on the training set is better than on the test set since the model was trained on the training set data. Generally, the prediction rate of disease state in the test set is a better indicator of model accuracy since the model was not trained on the test data.

The same split of the data into training and test set will be used for other classification methods.

8.4 Random Forests

In a random forest, many classification trees are estimated where different trees in the forest are forced to use certain subsets of the predicting features and samples. After all the trees have been produced, each tree gets a vote as to how new data should be classified. With this many-classifiers voting scheme, the random forest is called an *aggregate classifier*. By design, Random Forests are less dependent on just a handful of genes like we've seen with single classification trees, and thus tend to be more robust to spurious predictors in the training set that don't generalize well to new data. Also, although a random forest uses many different genes, the importance of each

gene in the overall aggregate classifier can be calculated and studied:

Example 1: Application to the Chiaretti (2004) data.

```
> Y <- factor(ALLBTnames$BT);X <- t(probeData)
> rf1 <- randomForest(X, Y, ntree = 1000, importance=TRUE, proximity=TRUE)
> rf1

Call:
 randomForest(x = X, y = Y, ntree = 1000, importance = TRUE, proximity = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 9

        OOB estimate of  error rate: 19.23%
Confusion matrix:
   B1 B2 B3 class.error
B1 13  4  2   0.3157895
B2  1 32  3   0.1111111
B3  0  5 18   0.2173913
> plot(rf1, main=NULL, lwd=3)
> legend("topright",c("OOB", "B1","B2","B3"),merge=TRUE,col=c(1,2,3,4),lty=c(1,1,1,1), lwd=
    ↪ c(4,4,4,4))
> varImpPlot(rf1,
+            n.var = 15,
+            pch=19,
+            main=NULL,
+            col="red",
+            gcolor="blue",
+            lcolor="darkgreen")
```

In Figure 8.11, we see from the `plot()` function the error rates for predicting the out-of-bag (OOB) samples as the random forest grows from 1 to 1,000 trees. Error rates are plotted for all the OOB samples (in black) as well for the three individual B-cell types. We can see that the error rates decrease as the forest gets bigger and that the forest has more difficulty correctly predicting the B1 and B3 subtypes. In Figure 8.10, we see the variable importance plot generated by `varImpPlot()`. We can see which variables are most important for proper classification. Note that the misclassification rate is $15/78 = 0.19$.

Example 2: Separate training and test sets. To evaluate the predictive accuracy, the random forest method is applied to the training and validation sets:

```
> rft <- randomForest(X[training,], Y[training], ntree = 1000, importance = TRUE, proximity
    ↪ =TRUE)
> table(rfpredt=rft$pred,Yt=Y[training])
       Yt
rfpredt B1 B2 B3
```

Error rates for predicting B-cell leukemia types for a Random Forest

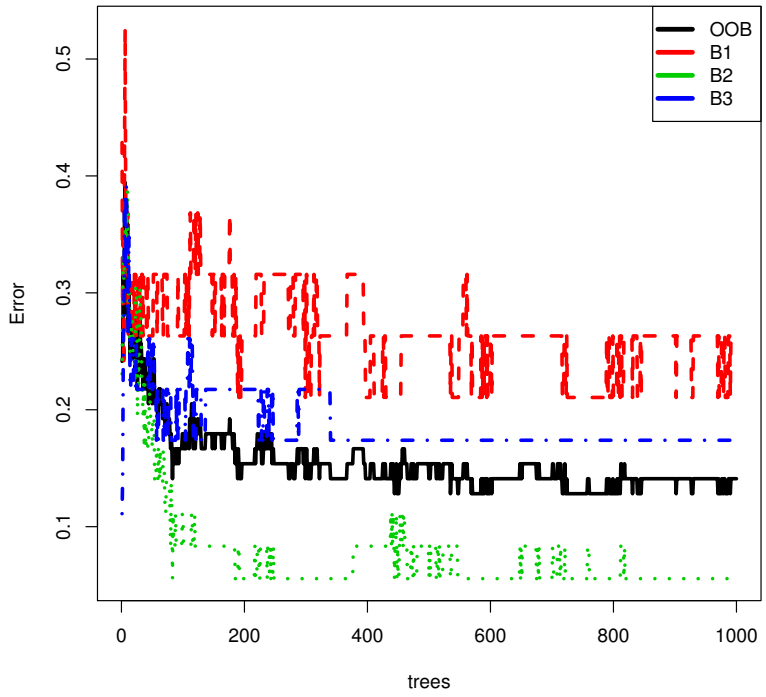

Figure 8.11: The error rates for predicting the out-of-bag (OOB) samples as the random forest grows from 1 to 1,000 trees. Error rates are plotted for all the OOB samples (in black) as well for the three individual B-cell types. We can see that the error rates decrease as the forest gets bigger and that the forest has more difficulty correctly predicting the B1 and B3 subtypes.

```
     B1 10  0  0
     B2  1 12  4
     B3  1  3  8
> pv <- predict(rft, newData=X[testing,], type="class", proximity=TRUE)
> table(rfpredv=pv$pred,Yv=Y[testing])
       Yv
rfpredv B1 B2 B3
     B1  5  1  0
     B2  1 15  4
     B3  2  4  7
```

The misclassification rate on the training set is $9/39 = 0.23$ and on the test set is $12/39 = 0.31$. The latter is worse than the model estimated by recursive partitioning.

8.5 Support Vector Machines

A support vector machine (SVM) finds separating lines (hyperplanes) with maximum margins between groups of points. However, before attempting to separate the data with a linear hyperplane, the data is first transformed via a kernel function ϕ into a higher dimensional space called the feature space. Therefore, when a nonlinear dividing line is needed in the original data space , instead of a fitting nonlinear curves to the data, SVM uses a kernel function ϕ to transform the data into a higher dimensional (feature) space where a linear hyperplane can be used to perform the separation. This separating hyperplane is then mapped back into the original data space (also called the input space).

Different kernel functions are used depending on the original data. A crucial step is selecting the best kernel that will transform the original data into a feature space where linear hyperplanes can properly separate the clusters. Many kernel mapping functions are available. However, a few kernel functions have been found to work well for a wide variety of biological applications: linear, radial basis function (RBF - default), polynomial, and sigmoidal kernel functions.

After transforming the data, if separating hyperplanes do exist in the feature space then a linear support vector machine will find them. This is because the optimization method in the feature space is based on quadratic programming methods to find the globally optimal solution. Support vector machines do not automatically select variables and are designed for continuous predictor variables. Since the mathematical details are beyond the current scope, we will focus on illustrating applications to gene expression data.

Example 1: Application to the Chiaretti (2004) data. The parameters for the support vector machine can be determined by the function `svm()` from the `e1071` package as follows:

```
> library(e1071)
```

```
> df <- data.frame(Y = factor(ALLBTnames$BT), X =t(probeData))
> Y <- factor(ALLBTnames$BT); X <- t(probeData)
> Yt <- factor(ALLBTnames$BT)[training]; Yv <- factor(ALLBTnames$BT)[testing]
> X <- t(probeData); Xt <- X[training,]; Xv <- X[testing,]
> svmEst <- svm(X, Y, data=df, type = "C-classification", kernel = "linear")
> svmPredict <- predict(svmEst, X, probability=TRUE)
> table(svmPredict, Y)
          Y
svmPredict B1 B2 B3
        B1 17  0  0
        B2  1 35  6
        B3  1  1 17
> summary(svmEst)

Call:
svm.default(x = X, y = Y, type = "C-classification", kernel = "linear",
    data = df)

Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.3333333

Number of Support Vectors:  38

 ( 17 8 13 )

Number of Classes:  3

Levels:
 B1 B2 B3
```

The confusion matrix shows that the misclassification rate of the three classes of B-cell ALL is 9/78=0.115, and thus the prediction is very good. However, note from `summary(svmEst)` that the number of support vectors per class are 17, 8, and 13 for classes B1, B2, and B3, respectively. These support vectors have values for all the input variables (genes) as can be obtained from `dim(svmEst$SV)` and the coefficient vectors `dim(svmEst$coefs)`:

```
> dim(svmEst$SV)
[1] 23  3
> dim(svmEst$coefs)
[1] 23  2
```

Hence, the excellent prediction properties are obtained through a very large number of estimated parameters.

Example 2: Separate training and test sets. A generally applied manner to evaluate the predictive quality of an estimated model is by splitting the data into a training and a test (validation) set. The model is estimated using the training set and then the class of the patients in the test set is predicted.

We shall use the same split as in Example 6 of the previous section:

```
> Yt <- factor(ALLBTnames$BT)[training]; Yv <- factor(ALLBTnames$BT)[testing]
> X <- t(probeData); Xt <- X[training,]; Xv <- X[testing,]
> svmEst <- svm(Xt, Yt, type = "C-classification", kernel = "linear")
> svmPredict.t <- predict(svmEst, Xt, probability=TRUE)
> table(svmPredict.t, Yt)
            Yt
svmPredict.t B1 B2 B3
          B1 12  0  0
          B2  0 13  1
          B3  0  2 11
> svmPredict.v <- predict(svmEst, Xv, probability=TRUE)
> table(svmPredict.v, Yv)
            Yv
svmPredict.v B1 B2 B3
          B1  6  3  1
          B2  1 18  4
          B3  0  0  6
```

The predictions of the disease states of the patients from the training set almost perfectly match the diagnosed states with an error rate of $3/39 = 0.077$. However, the predictions of the classes of the patients from the test set have a misclassification rate of $9/39 = 0.23$. Hence, the parameter estimates from the training set are sample specific and do not generalize with the same accuracy to the test set.

8.6 Neural Networks

Neural networks are nonlinear models that consist of nonlinear hyperplanes (hypersurfaces) around classes of objects (Ripley, 1996). A popular subclass of neural networks are the multilayer perceptrons (MLPs) that consist of one layer of (artificial neuronal) input nodes, one layer of output nodes, and one or more layers of hidden nodes in between the input and output nodes. Each layer of nodes is fully connected to the preceding and subsequent layer of nodes, whereby each node of the preceding layer are connected to all the nodes of the subsequent layer. Also, each node has a linear or nonlinear activation function that is trained via supervised learning techniques. With the right connected topology, neural networks can be great classifiers. However, the models themselves are difficult to interpret - so typically little biological insight is acquired. Typically, the nonlinear models are trained using an iterative backpropagation method that minimizes the classification error rate with the set of predictor variables. We will illustrate two applications of neural networks using the Chiaretti et al. 2004 data.

Example 1: Application to the Chiaretti (2004) data. The models can be estimated with the `nnet()` function from the `nnet` package. To avoid having too many variables, we randomly select a subset of 20 genes:

```
> Y <- factor(ALLBTnames$BT); X <- t(probeData)
> library(nnet)
> df <- data.frame(Y = Y, X = X[, sample(ncol(X), 20)])
> nnest <- nnet(Y ~ .,data = df, size = 5, maxit = 500, decay = 0.01, MaxNWts = 5000)
> pred <- predict(nnest, type = "class")
> table(pred, Y) # prints confusion ma
    Y
pred B1 B2 B3
  B1 19  0  0
  B2  0 36  0
  B3  0  0 23
```

The confusion matrix shows that zero out of 78 patients are misclassified.

Example 2: Separate training and test sets. The results from using our training and test data on the neural networks are as follows:

```
> nnest.t <- nnet(Y ~ ., data = df,subset=training, size = 5,decay = 0.01,maxit=500)
converged
> prednnt <- predict(nnest.t, df[training,], type = "class")
> table(prednnt,Ytrain=Y[training])
       Ytrain
prednnt B1 B2 B3
     B1 12  0  0
     B2  0 12  1
     B3  0  3 11
> prednnv <- predict(nnest.t, df[testing,], type = "class")
> table(prednnv, Yval= Y[testing])
       Yval
prednnv B1 B2 B3
     B1  6  2  1
     B2  1 19  4
     B3  0  0  6
```

The predictions on the training set have a misclassification rate of $4/39 = 0.10$ and that on the test set $8/39 = 0.21$.

8.7 Logistic Regression with Generalized Linear Models

Within the framework of generalized linear models (GLMs), the diagnosis of a patient is seen as a response. When the response has the values healthy or diseased, it may be assumed that the binomial distribution holds with a success probability p_i. Recall from Chapter 3 that for a binomially distributed

variable, the probability of y_i successes out of n_i trials is given by:

$$P(Y_i = y_i) = \frac{n_i!}{y_i!(n_i - y_i)!} p_i^{y_i}(1 - p_i)^{n_i - y_i}, \qquad \text{for} \quad k = 0, \cdots, n_i.$$

The value of p_i is closely related to one or more predictor variables x_1 and x_2 via a linear combination. That is, the linear model holds such that $\eta_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}$. The predictors are linked to the success probability via the so-called *logit* link:

$$p_i = \frac{e^{\eta_i}}{e^{\eta_i} + 1} = \frac{\exp(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2})}{1 + \exp(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2})}.$$

Rather than going deeper into the details, the usefulness of generalized linear models will be illustrated with two examples.

Example 1: Classification with CCND3 gene expression. In the Golub et al. (1999) data, we may label a patient Y_i as $Y_i = 1$ if the patient is diagnosed with ALL and $Y_i = 0$ if (s)he is diagnosed with AML. We will use the CCND3 (Cyclin D3) gene expression values as our predictor variable. Also, we will compute the response with the formula `-golub.cl + 1`. This yields 1 for ALL and 0 for not ALL[3]. We will use the `glm()` function with the *logit* link function to perform the logistic regression:

```
> library(faraway)
> ccnd3 <- grep("CCND3",golub.gnames[,2], ignore.case = TRUE)
> logitmod <- glm((-golub.cl + 1) ~ golub[ccnd3,],family=binomial(link = "logit"))
> pchisq(deviance(logitmod),df.residual(logitmod),lower=FALSE)
> plot((-golub.cl + 1) ~ golub[ccnd3,],
+ xlim=c(-2,5),
+ ylim = c(0,1),
+ xlab="CCND3 expression values ",
+ ylab="Probability of ALL")
> x <- seq(-2,5,.1)
> lines(x,ilogit(-4.844124  + 4.439953*x))
> pchisq(deviance(logitmod),df.residual(logitmod),lower=FALSE)
> summary(logitmod)

Call:
glm(formula = (-golub.cl + 1) ~ golub[ccnd3, ], family = binomial(link = "logit"))

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)     -4.844      1.849  -2.620  0.00880 **
golub[ccnd3, ]   4.440      1.488   2.984  0.00284 **
```

[3] One may also conveniently use a factor as a response variable

```
    Null deviance: 45.728  on 37  degrees of freedom
Residual deviance: 18.270  on 36  degrees of freedom
AIC: 22.270
```

From Figure 8.12, we can see that the logit curve fits the data fairly well. From the summary of the output we can see that the estimated intercept is -4.844 and the estimated slope is 4.440. Both coefficients are significantly different from zero. The goodness-of-fit value of the model is computed from the chi-square distribution and equals .99. Therefore, the model fits the data well. The predictive accuracy of the model can be obtained using the `predict()` and `table()` functions:

```
> predictor <- predict(logitmod,type="response") > 0.5
> predict.factor <- factor(predictor,levels=c(TRUE,FALSE),labels=c("ALL","not ALL"))
> table(predict.factor,golubFactor)
            golubFactor
predict.factor  ALL AML
  ALL        26   2
  not ALL     1   9
```

The diagnosis of the majority of patients is predicted correctly.

Example 2: Application to the Chiaretti (2004) data. With the ALL data, we want to model the diagnosis of B-cell State B1, B2, or B3 as a response. The factor representing these levels can be used as input for the response. Here we use the gene expressions with greatest importance from the classification tree in Section 8.3. We assign the biological name to the predictor variables and estimate the generalized linear model (GLM):

```
> library(nnet); library("hgu95av2.db"); library(ALL);data(ALL)
> probe.names <- c("1389_at","35991_at","40440_at")
> ALLB123 <- ALL[,ALL$BT %in% c("B1","B2","B3")]
> probeData <- exprs(ALLB123)[probe.names,]
> row.names(probeData) <- unlist(mget(probe.names, env = hgu95av2SYMBOL))
> factor <- factor(ALLB123$BT,levels=c("B1","B2","B3"))
> data <- data.frame(factor,t(probeData))
> mnmod <- multinom(factor ~ ., family=binomial(link = "logit"),data=data)
> summary(mnmod)
Call:
multinom(formula = factor ~ ., data = data, family = binomial(link = "logit"))

Coefficients:
   (Intercept)     MME       LSM6    SERBP1
B2    14.36158 4.14002 -0.8494635 -5.104337
B3   -12.90584 4.94908  4.7415802 -5.655420

Std. Errors:
   (Intercept)      MME     LSM6   SERBP1
B2    16.36959 1.367058 1.716513 2.222486
B3    17.97896 1.424526 1.744425 2.313700

Residual Deviance: 59.88298
AIC: 75.88298
```

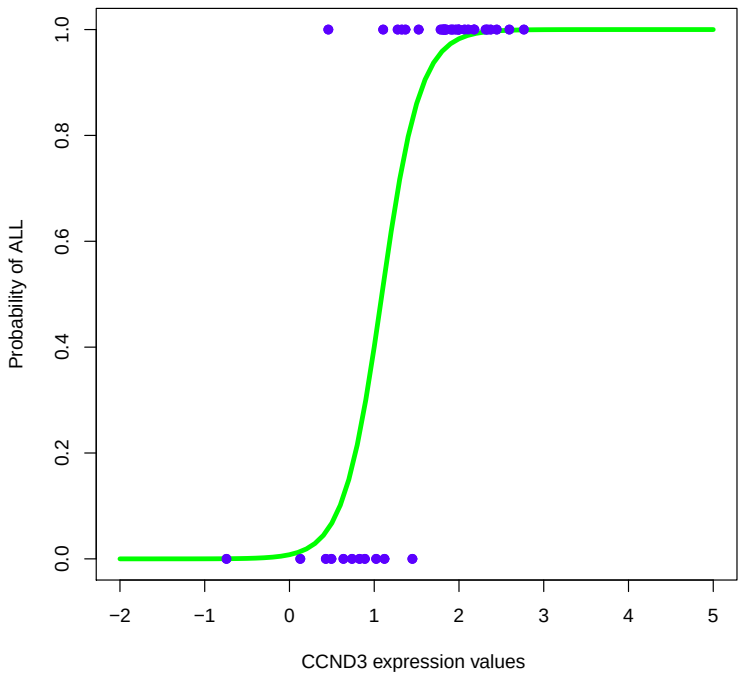

Figure 8.12: Logit fit to the CCND3 (Cyclin D3) expression values for the probability of having ALL vs AML.

Apart from the intercepts, the estimated coefficients are significantly different from zero.

```
> predict.mn <- predict(mnmod,type="class")
> table(predict.mn,factor)
          factor
predict.mn B1 B2 B3
    B1 17  2  1
    B2  1 31  5
    B3  1  3 17
```

The model predicts the diagnosed classes quite well with a misclassification rate of $13/78 = 0.17$.

Generalized linear models (GLMs) are statistical models which have to

be estimated by an iterative method. GLMs have the advantage that confidence intervals and the significance of the parameters can be estimated. For large amounts of predictor variables, additional terms are often added to the regression model to penalize for the number of parameters - and are thus called penalized or regularized regression models.

8.8 Overview and concluding remarks

Central themes in classification methods are the clarity of the model, the size of the model, and the predictive accuracy of the model (on a test set). For many researchers, it is of crucial importance to have a clear idea of what a method is essentially doing. Some models and their estimation procedures are mathematically complicated and are often treated by many researchers as black boxes (e.g. Neural Nets). From a more pragmatic point of view, methods that provide models that are difficult to interpret may still be of value if predictive accuracy is all that matters. However, support vector machines (SVMs) and neural networks (NNs) typically use a large number of parameters to predict well on the training set, but less well on a test set. Therefore, one should be alert that excellent predictive accuracy on the training set may in fact be indicative of overfitting and seriously compromise accuracy on new data.

Recursive partitioning to estimate a classification tree performs very well on variable selection and pruning in order to discover as few variables (e.g. gene expressions) as possible for maximum predictive accuracy. In addition, classification trees have great clarity. See the CART package (Breiman et al., 1984) for further types of recursive trees. Note that several methods have different misclassification rates with respect to the whole sample, but comparable rates on the test sets. It should, however, be clear that when there are non-linear relationships between predictor variables and classes, then nonlinear models should outperform linear models[4].

8.9 Exercises

1. **Classification trees with the ALL data.** Perform recursive partitioning with the `rpart()` function. Like Example 5, using the ALL

[4]Some people may want to use the `ade4TkGUI()`

data, we want to predict the diagnosis of B-cell state B1, B2, or B3 from the gene expressions.

(a) Select the genes with different means over the patient groups with an ANOVA p-value smaller than 0.001.

(b) Using `rpart()`, find a manner to identify the best 2 genes from (a) to predict the B-cell state.

(c) Use `rpart()` again to construct the best overtall classification tree from the set of genes from (a) to predict the B-cell state.

(d) How do the trees and misclassification rates differ?

2. **Sensitivity versus specificity.**

(a) Produce a sensitivity versus specificity plot for the gene expression values of CCND3 (Cyclin D3).

(b) In what sense does the plot resemble Figure 8.1?

(c) Compute the area under the curve for the sensitivity versus specificity curve.

3. **Comparing Classification Methods.** To obtain an idea on the misclassification rates when there is no actual relation between the predictors and the factor, we will perform a small simulation study.

(a) Construct a factor consisting of 50 ones and then 50 twos, and a 100 rows by 4 columns matrix with predictor variables with values from the normal distribution $\mathcal{N}(0, 1.0^2)$. Use the four letters "A", "B", "C", and "D" for the column names.

(b) Use `rpart()` to construct a recursive tree and report the misclassification rate. Comment on the results.

(c) Do the same for a support vector machine classifier applied to the data.

(d) Do the same for a neural network classifier applied to the data.

(e) Think through your results and comment on them.

4. **Prediction of achieved remission.** In the ALL data, the patients are checked for achieving remission. Specifically, the variable `ALL$CR` has values CR (became healthy) and REF (did not respond to therapy; remained ill).

(a) Construct an expression set containing the patients with values for the phenotypical variable `remission` and the genes with expressions with a significant p-value for the t-test that rejects the hypothesis of equal means between the groups CR or REF.

(b) Use recursive partitioning to predict the remission using the above data set. Report the misclassification rate and the names of the genes that play a role in the tree.

5. **Gene selection by AUROC.** A strategy for selecting genes is to compute the AUC for each gene and to use the best 10 for further investigation. Compute the AUC for each row of gene expressions in the Golub at al. (1999) data. Collect the AUCs in a vector and select the ten best. Is CCND3 (Cyclin D3) among these?

6. **Classification Tree for E. coli.** The E. coli data can be downloaded with the following code:

```
> ecoli = read.table("http://www.grappa.univ-lille3.fr/~torre/Recherche/Datasets/
    ↪ downloads/ecoli/ecoli.data",sep=",",header = TRUE)
> colnames(ecoli) <- c("SequenceName","mcg","gvh","lip","chg","aac","alm1","alm2","
    ↪ ecclass")
```

(a) Use the `ecclass()` function to construct a factor containing the "cp", "im", and "pp" levels.

(b) Construct a classification tree using the variables "mcg", "gvh", "lip", "aac", "alm1", "alm2" to predict the 3 different levels (classes). Give the code. Hint: Use the addition notation.

(c) Plot the tree and report the variables that play a role in the classification tree.

(d) Predict all the classes using the tree. Report the misclassification rate.

(e) Leave out the best predictor variable in the classification tree and reconstruct the tree. Report the new misclassification rate. Is it much worse?