2 Tumor classification by SVM (Projekt 2)

2.1 Introduction

How many patients are in the full data set? 128

How many are labelled BCR/ABL or NEG? 111

and how many have B-cell ALL? 95

How many of those that have B-cell ALL are labelled BCR/ABL or NEG? 79

What would happen if you didn't update the labels for the subset of data? You would have several unused levels in the factor, so you wouldn't get the two class-labels 0 and 1 (which are expected by `mt.teststat`)

How many patients are in the training set? 73

How many of them are labelled BCR/ABL? 35

and how many are labelled NEG? 38

2.2 Support Vector Machines (SVMs)

Compute the variance for each gene over all patients, and plot the results as a histogram The variances (for 12625 genes) are calculated and stored in the variable "train.var":

```
> head(train.var)
   1000_at     1001_at   1002_f_at   1003_s_at     1004_at     1005_at
0.06912196 0.10384373 0.03595022 0.07002139 0.06870848 1.24412290
```

See Figure 1 for the plot as a histogram.

To speed-up computation reduce the number of genes in the data by focusing on the 1000 genes with greatest variance The data (for the top 1000 genes) is stored in the variables `train.top1k` (gene names), `train.top1k.data` (ExpressionSet) and `train.top1k.e` (matrix of expression values)

```
> head(train.top1k)
[1] "36873_at" "36633_at" "36753_at" "40647_at" "40661_at" "41401_at"
> print(train.top1k.data)
ExpressionSet (storageMode: lockedEnvironment)
assayData: 1000 features, 73 samples
(...)
> str(train.top1k.e)
 num [1:73, 1:1000] 3.36 3.47 3.13 3.23 4.18 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:73] "01005" "01010" "03002" "04007" ...
  ..$ : chr [1:1000] "36873_at" "36633_at" "36753_at" "40647_at" ...
```

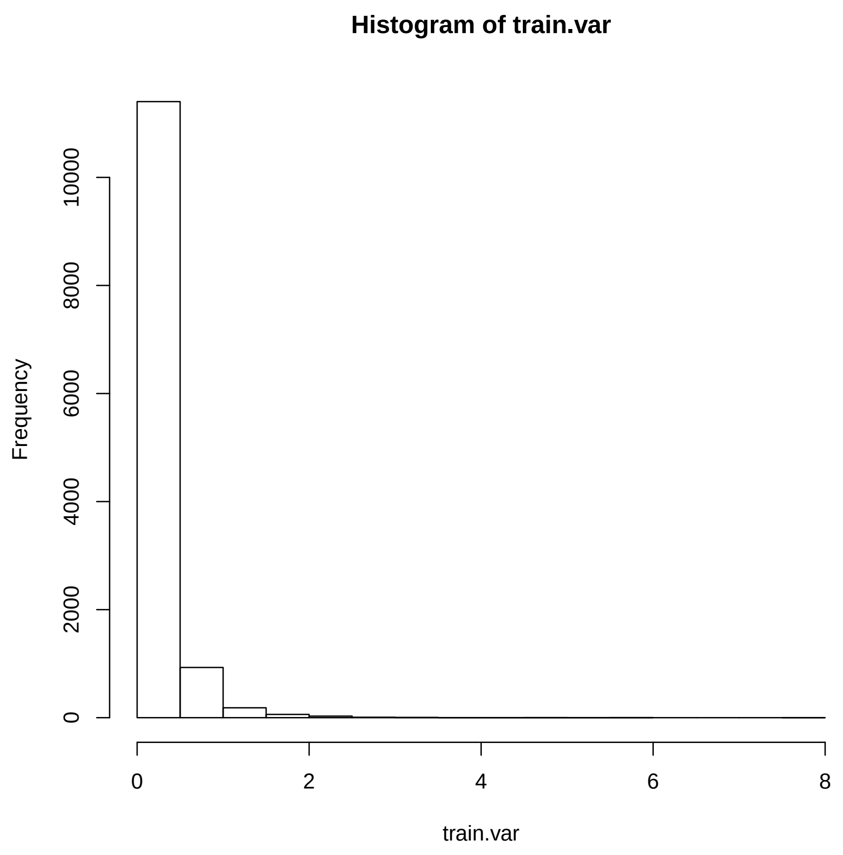

Figure 1: histogram of the variance for each gene over all patients

Use the function summary to get an overview of the model produced by the SVM and to check that you really did perform a classification rather than a regression task

```
Call:
svm.default(x = train.top1k.e, y = train.top1k.data$label, kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.001

Number of Support Vectors:  48

 ( 22 26 )


Number of Classes:  2

Levels:
 BCR/ABL NEG
```

Use the method predict on the training-set using the results of the previous command and tabulate the results using table

```
train.pred BCR/ABL NEG
   BCR/ABL      34   0
   NEG           0  39
```

How many errors do you have? The prediction on the training-set gives perfect results, there are no errors.

Again use the function summary to get an overview of the model produced by the SVM (but this time set the parameter so that a 10-fold cross validation occurs)

```
Call:
svm.default(x = train.top1k.e, y = train.top1k.data$label, kernel = "linear",
    cross = 10)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.001

Number of Support Vectors:  48

 ( 22 26 )


Number of Classes:  2

Levels:
 BCR/ABL NEG

10-fold cross-validation on training data:

Total Accuracy: 83.56164
Single Accuracies:
 71.42857 100 71.42857 87.5 85.71429 71.42857 87.5 85.71429 85.71429 87.5
```

Why does the line "Single Accuracies" change each time? The "Single Accuracies" change due to the way cross-validation works: The list of patients is randomly split into 10 subsets, so each time other patients are used to calculate the accuracy

How does the total accuracy change? The "Total Accuracy" is the mean of the "Single Accuracies", so it will fluctuate a few percent around a mean every time.

What does the SVM tell you about the six new patients?

```
test.pred BCR/ABL NEG
  BCR/ABL       3   2
  NEG           0   1
```

The prediction on the test-set gives worse results than on the training-set, there are some errors. So a training-error of zero does not guarantee a good prediction.

Generate a randomly ordered vector of labels and train a model using a linear kernel SVM

```
Call:
svm.default(x = rand.e, y = rand.data$label, kernel = "linear", cross = 10)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.001

Number of Support Vectors:  64

 ( 30 34 )


Number of Classes:  2

Levels:
 BCR/ABL NEG

10-fold cross-validation on training data:

Total Accuracy: 58.90411
Single Accuracies:
 28.57143 71.42857 85.71429 75 42.85714 57.14286 75 57.14286 57.14286 37.5
```

What do you notice about the prediction results? What do you notice about the CV error between the original and your random data.

```
# prediction on training-data:
rand.pred BCR/ABL NEG
  BCR/ABL      34   0
  NEG           0  39

# prediction on test-data:
rand.pred BCR/ABL NEG
  BCR/ABL       2   2
  NEG           1   1
```

If any biological information in the data is destroyed by randomly generating the labels the prediction on the training-set still gives perfect results without errors, but the total accuracy decreases and the prediction on the test-set gives a lot of errors.

Write a function for feature selection using the t-statistic: Function `informative_genes` takes a matrix of expression values (`data`), a vector of labels (`labels`) and a number of genes (`nrGenes`) and returns a vector of indices of informative genes.

```
informative_genes <- function(data, labels, nrGenes)
    which(order(mt.teststat(data, as.numeric(factor(labels))-1, test="t"),
        decreasing=TRUE) <= nrGenes)

> top10 <- informative_genes(exprs(subset.data), subset.data$label, 10)
> top10
 [1]   787  1063  4530  4845  5676  6035  8925  9867 12154 12359
> rownames(exprs(subset.data)[top10,])
 [1] "1702_at"    "1957_s_at"  "34488_i_at" "347_s_at"   "35622_at"
 [6] "35978_at"   "38841_at"   "39774_at"   "590_at"     "797_at"
```

Select the 100 genes with the largest t-statistic: The data (for the top 100 genes) is stored in the variables `train.top100` (gene indices), `train.top100.data` (ExpressionSet) and `train.top100.e` (transposed matrix of expression values)

Train a SVM on this reduced data-set with different kernels and parameters and compare the results to those obtained with all genes. Do cross-validation on the reduced data-set. Vary the number of selected genes. How many genes do you need to still get a reasonable CV error?

```
> accuracy
     polynomial  sigmoid   linear   radial
1      43.83562 47.94521 45.20548 45.20548
2      47.94521 47.94521 42.46575 46.57534
10     52.05479 63.01370 63.01370 57.53425
50     42.46575 49.31507 50.68493 47.94521
75     45.20548 53.42466 64.38356 54.79452
100    43.83562 57.53425 64.38356 57.53425
150    42.46575 69.86301 68.49315 63.01370
200    49.31507 60.27397 67.12329 63.01370
500    42.46575 61.64384 67.12329 60.27397
1000   43.83562 72.60274 75.34247 63.01370
```

With the “linear”-kernel the total accuracy is usually higher than 60% when using at least 75 genes (see also Figure 2)

2.3 Model assessment

Explain why selecting genes with large variance over all the samples does not result in the same distortion of cross validation as feature selection by t-scores. Feature selection by t-scores

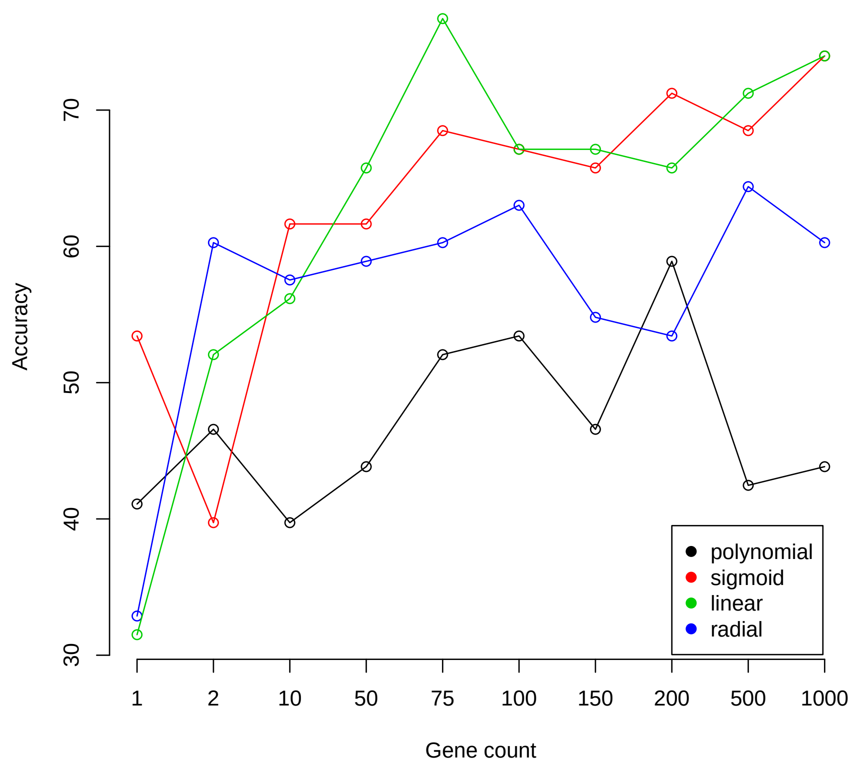

Figure 2: Total accuracies for various kernels and different numbers of selected genes

takes into account that there are different classes of patients ("labels": NEG or BCR/ABL) which selection by variance does not, so other genes are used which results into a different distortion of cross validation.

Write your own function for 10-fold cross-validation with "in-loop" feature selection: Function `cross_validate` takes a matrix of expression values (`data`), a vector of labels (`labels`), the number of cross-validation-steps (`steps`) and a number of genes to select in each step (`nrGenes`) and returns a list containing the total accuracy (`$tot.accuracy`) and the vector of single accuracies (`$accuracies`).

```
> cross_validate(rand.e, rand.data$label, 10, 100)
$accuracies
 [1] 25.00000 50.00000 75.00000 57.14286 57.14286 57.14286 28.57143 28.57143
 [9] 42.85714 71.42857

$tot.accuracy
[1] 49.28571
```

Try cross validation with "out of the loop" feature selection and with "in-loop" feature selection at least 100 times, gather the results in two vectors and compare the distribution of results

```
> t.test(in_loop, out_loop)

        Welch Two Sample t-test

data:  in_loop and out_loop
t = -3.6788, df = 159.157, p-value = 0.0003203
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.2817705 -0.9889732
sample estimates:
mean of x mean of y
 50.28929  52.42466
```

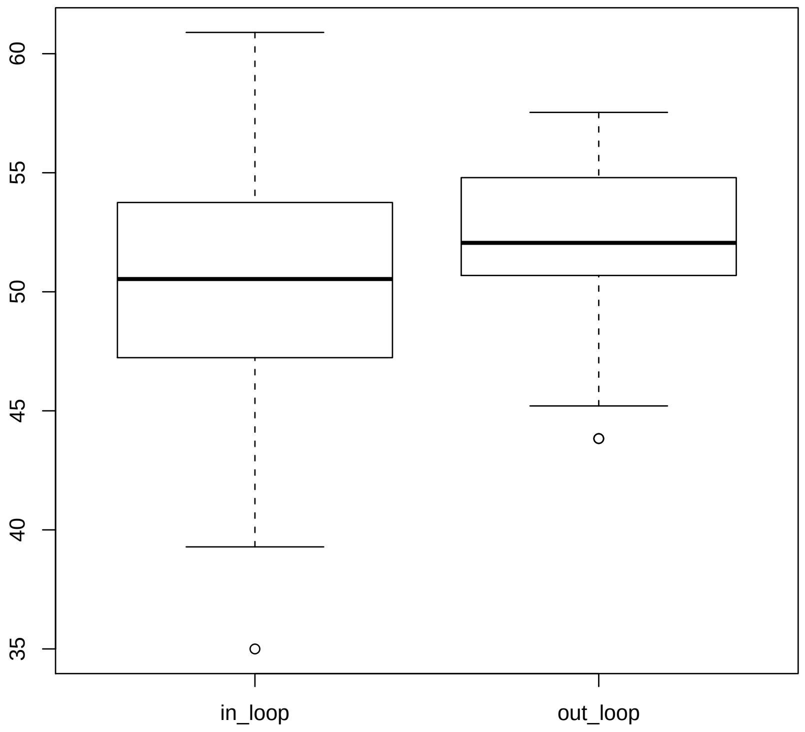

Figure 3: Boxplot "in-loop" vs. "out of the loop" accuracies

Explain why the variance of the results is greater when using cross-validation with "in-loop" feature selection: The variance of the results is greater when using cross-validation with "in-loop" feature selection, because for each cross-validation step a different set of genes is used, which results in a higher total variance.

Write a nested-loop CV and plot a vote matrix that visualizes how often the samples are misclassified in cross-validation: Using the `MCRestimate`-package simply call the function `MCRestimate` with the supplied parameters:

```
nl_cross_validate <- function(known.patients, class.column,
    classification.fun, poss.parameters = list(),
    cross.outer = 10, cross.repeat = 3, cross.inner = cross.outer)
{
    result <- MCRestimate(eset=known.patients,
        class.column=class.column,
        classification.fun=classification.fun,
        poss.parameters=poss.parameters,
        cross.outer=cross.outer,
        cross.repeat=cross.repeat,
        cross.inner=cross.inner)
    plot(result)
}

> nl_cross_validate(top100.data, "label", "SVM.wrap")
```

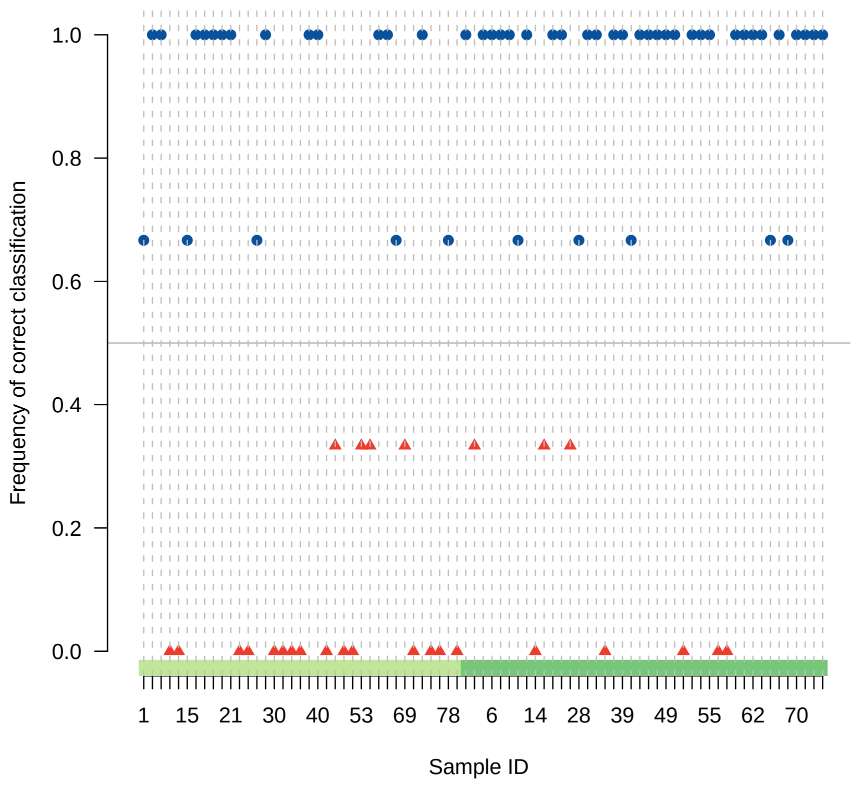

Figure 4: Vote matrix that visualizes how often the samples are misclassified in cross-validation (of the 100 genes with the largest t-statistic)