Exercise Classification

Hubert Rehrauer

November 28, 2016

1 Loading an example data set

In Bioconductor there is an example data set from the publication:

Sabina Chiaretti, Xiaochun Li, Robert Gentleman, Antonella Vitale, Marco Vignetti, Franco Mandelli, Jerome Ritz, and Robin Foa. *Gene expression profile of adult T-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival.* **Blood**, 1 April 2004, Vol. 103, No. 7.

The data set contains 128 samples that were used to characterize subtypes of acute lymphoblastic leukemia.

```
> library(ALL)
> data(ALL)
> show(ALL)
```

In this exercise we will only use samples from B-cells and the molecular biological characterizations "BCR/ABL" and "NEG".

```
>  bCellSamples = grep("^B", ALL$BT)
>  BcrAndNegSamples = which(ALL$mol.biol %in% c("BCR/ABL", "NEG"))
>  samplesToUse = intersect(bCellSamples, BcrAndNegSamples)
>  dataMatrix = exprs(ALL[ ,samplesToUse])
>  classLabels = factor(ALL$mol.biol[samplesToUse])
```

Now we have our data as a matrix in `dataMatrix` for which the rows are the genes which will be used as features (variables) to classify the columns which represent the samples. The dimension of the matrix is

```
>  dim(dataMatrix)
```

```
[1] 12625    79
```

2 Prefiltering of Genes

In order to reduce the dimensionality, do select only the 1000 genes with the highest variance in our data set. Remove the low variance genes from the data set:

```
>   highVarGenes = ...
>   dataMatrix = ...
```

3 Support Vector Machine Classification

Load the necessary library:

```
> library(e1071)
```

The model is generated by (note that the matrix has to be transposed):

```
>   model = svm(t(dataMatrix), classLabels, kernel = "linear")
```

and can then be used to predict the training samples

```
> predicted = predict(model, t(dataMatrix))
> table(true = classLabels, pred = predicted)
```

```
         pred
true      BCR/ABL NEG
  BCR/ABL      37   0
  NEG           0  42
```

and shows that the training error is zero.

Now, we run a cross validation to estimate the error when classifying unknown samples

```
> model.cv = svm(t(dataMatrix), classLabels, kernel = "linear", cross = length(classLabels
> summary(model.cv)
```

the average rate of correct classification is given by

```
> model.cv$tot.accuracy
```

```
[1] 81.01266
```

Now use the knn-classifier with $k = 5$ to run leave-one-out cross-validation (see the help for `knn.cv`). The method returns the predicted label for each of the samples. Compute the accuracy (the percentage of samples for which the predicted label is equal to the true label).

4 Top Scoring Pairs Classification

```
> library(tspair)
```

They will be installed in a local directory and only be avaialable for the current user.

Load the library (tspair) and find the top-scoring pair with the function (tspcalc).

```
>   tspResult = tspcalc...
```

Compare the predictions with the true class labels

```
>  predictedLabels = predict(tspResult, dataMatrix)
>  table(predictedLabels, classLabels)
```

```
               classLabels
predictedLabels BCR/ABL NEG
        BCR/ABL      30   4
        NEG           7  38
```

Now implement yourself a leaf-one out cross-validation to find a more realistic estimate of the accuracy. Use a (for) loop with

```
>   for( i in 1:ncol(dataMatrix)){
+    ...
+   }
```

5 Parameter Optimization

Find the value for k for which the classifier works best. Use the method `tune.knn` to scan the value range from 1 to 20.

Use the optimal value for k to classify the unknown samples with the method `knn`