

# R/BioC Exercises: Supervised Learning

Emiel Ver Loren van Themaat and Perry Moerland

April 22, 2010

☞ This document and information on how to log on to a PC in the exercise room and the UNIX server can be found here:

<http://bioinformaticslaboratory.nl/twiki/bin/view/BioLab/EducationBioinformaticsII>

☞ For this exercise you will need codes written in the book (Chapter 17). You can retrieve the digital version of the book codes from the site: [http://www.bioconductor.org/mogr/chapter-code/Computational\\_Inference.R](http://www.bioconductor.org/mogr/chapter-code/Computational_Inference.R)

## Introduction

First, we will look at the behaviour of some classification models on a two-dimensional dataset. We will use functions from the package `MLInterfaces`. Useful chapters for this lab are Chapters 16, 17, and 24.

📖 In the package `MLInterfaces`, classification functions from all kinds of packages are interfaced. All these classification functions have a single `MLInterfaces` interface which always has a common set of arguments, besides the model specific arguments. The names of these interface methods have a 'I' appended to the name of the original function. For example, *randomForest* has a `MLInterfaces` equivalent *randomForestI*. For the set of methods included, type:

```
> help(MLearn)
```

The data we will use comes from one of the early high-profile microarray papers of Golub *et al* ([http://www-genome.wi.mit.edu/mpr/data\\_set\\_ALL\\_AML.html](http://www-genome.wi.mit.edu/mpr/data_set_ALL_AML.html)). The experiment consisted of 47 patients with acute lymphoblastic leukemia (ALL) and 25 patients with acute myeloid leukemia (AML). The samples were assayed using Affymetrix hgu6800 chips and data on the expression of 7129 genes (Affymetrix probesets) are available. Data is available as an *ExpressionSet* from the package `golubEsets`.

You will need to load the following packages after starting R 2.10:

```
library(MLInterfaces)
library(randomForest)
library(class)
library(e1071)
library(Biobase)
library(golubEsets)
```

- Exercise 1:**
- Extract the Golub dataset with `data(Golub_Merge)` and inspect the resulting object.
  - Select only the part of the *ExpressionSet* corresponding to the probesets with index 7000 and 7001. Now use this subset to train the following classifiers that you encountered in the lecture: *ldaI*, *nnetI*, *svmI*, and *rpartI*. Use the first 35 samples as training data. Which classifiers do these functions correspond to?
  - Use the function *planarPlot* to show the classification boundaries of the four models. Can you explain the shape of the boundaries for each of them?
  - Plot the points from the training set in the planar plots you made before and give them a class-specific color (ALL/AML). Now repeat the previous steps while changing some of the parameters of the classification routines: for example, the number of hidden units in a neural network or the type of kernel in a support vector machine. Try to explain the influence of each of the changes.

## SELDI-TOF ovarian cancer dataset

We will now start using a dataset generated by a proteomics experiment. Instead of measuring RNA abundances using microarray platforms, protein abundances are measured directly using SELDI-TOF<sup>1</sup>.

Blood samples from 14 controls and 14 patients with ovarian cancer are used in our experiment. On the basis of the measured protein profiles we hope to be able to diagnose new patients. The resulting, preprocessed, normalized data contains about 200 measurements per patient. Large parts of Chapter 17 will be used to see if we can create a *reliable* diagnostic profile. Remember everything you heard about possible biases in the classification lectures.

### Small trees as learners

Load the dataset `proteomics.RData` from the folder `/data/home/stud00/ML` into your R workspace. Check the class and attributes (*pData*) of the loaded object.

In Chapter 17.6.1 an example of evaluating classification performance is given for the ALL experiment. Four models are tested for their performance using bootstrapping (sampling with replacement). For each bootstrap sample, the top 100 genes associated with the outcome based on the bootstrapped training sample are selected by function *var\_selection*. In the random forest classification model, an ensemble of 500 decision trees is built for each bootstrap sample. Each tree is designed on  $n$  samples, where  $n$  is equal to the number of cases in the training set. The random forest procedure is used with two ways of selecting candidate genes from the top 100 genes: random sampling of three genes for each node in the trees of the forest and without random sampling. The latter is better known as *bagging*. The other procedure, logitboost, uses boosting together with “decision stumps” (trees with only one node) as weak learners. The 4th and last procedure is just guessing. The results on their data are shown in Figure 17.1.

---

<sup>1</sup>For more information on SELDI-TOF, check for example <http://en.wikipedia.org/wiki/SELDI-TOF>.

**Exercise 2:** • Try to use the same procedure used to create Figure 17.1 for our dataset but adapt it to the recent version of `MLInterfaces` (that is, use `MLearn`) and replace the `logitboostB` function with the a non-ensemble classifier `knnI` (make sure you adjust the arguments). Note that for the current version of `MLInterfaces`, you need to replace all occurrences of `something@predLabels` by `testPredictions(something)`.

- Do you think classification accuracy of the three non-guessing methods is better than for guessing (use the appropriate statistical test)?
- Is there a classifier which outperforms the others (use the appropriate statistical test)?
- What is the difference between specificity and the sensitivity, and which one is more important for this experiment?

**Exercise 3:** In the latest version of `MLInterfaces` `logitboostB` is missing. The package `caTools` contains a similar function `LogitBoost`. Function `predict.LogitBoost` can be used to predict outcomes for a test set based on a trained model. Try to use these functions directly and incorporate them into your code as a 5th model. Set `nIter` to 100.

**Exercise 4:** Adapt your code to retrieve the ten most selected features over the 100 bootstrap iterations.

```
> sessionInfo()
```

```
R version 2.10.1 (2009-12-14)
i386-pc-mingw32
```

```
locale:
```

```
[1] LC_COLLATE=English_United Kingdom.1252
[2] LC_CTYPE=English_United Kingdom.1252
[3] LC_MONETARY=English_United Kingdom.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United Kingdom.1252
```

```
attached base packages:
```

```
[1] stats      graphics  grDevices  utils      datasets  methods
[7] base
```