## HOW TO use MCRestimate

### Markus Ruschhaupt

March 22, 2005

## Estimation the misclassification error

Every classification task starts with data. Here we choose the well known ALL/AML data set from T.Golub (1) availabel in the package *golubEsets*. Furthermore, we specify the name of the phenodata column that should be used for the classification.

```
> library(MCRestimate)
> library(randomForest)
> library(golubEsets)
> data(golubTrain)
> class.colum <- "ALL.AML"</pre>
```

Cross-validation is an appropriate instrument to estimate the misclassification rate of an algorithm that should be used for classification. Often a variable selection or aggregation is applied to the data set before the main classification porcedure is started. But these steps must also be part of the cross-validation. In our example we want to perform a variable selection and only take the genes with the highest variance across all samples. Because we don't know exactly how many genes we want to have, we give two possible values (250 and 1000). The described methods is implemented in the functions g.red.highest.var. Further preprocessing functions are part of the package MCRestimate and also new functions can be implemented.

To use MCRestimate with a classification procedure a wrapper for this method must be available. The package MCRestimate includes wrapper for the following classification functions.

**RF.wrap** wrapper for random forest (based on the package randomForest)

**PAM.wrap** wrapper for PAM (based on the package pamr)

**PLR.wrap** wrapper for the penalised logistic regression (based on the package *MCRestimate*)

**SVM.wrap** wrapper for support vector machines (based on the package e1071)

**GPLS.wrap** wrapper for generalised partial least squares (based on the package *gpls*)

It is easy to write a wrapper for a new classification method. Here we want to use random forest for our classification task.

```
> class.function <- "RF.wrap"</pre>
```

Parameter to optimize: Most classification and preprocessing methods have parameters that must be choosen and/or optimized. In our example we choose two possible number of clusters for our preprocessing method. In each cross-validation step MCRestimate will choose the value that have the lowes misclassification error on the training set. This will be estimated through a second (inner) cross-validation with the function tune available in the package e1071.

Parameter for the plot of the results (see man page for details): We want to have the sample names as labels in the resulting plot. Samples is the name of the phenoData column the sample names are stored in.

```
> plot.label <- "Samples"</pre>
```

The Cross-validation parameter (see man page for details): For time reasons here we choose very small values. Normally the values for *cross.outer* and *cross.inner* should be at least 5.

```
> cross.outer <- 2
> cross.repeat <- 3
> cross.inner <- 2</pre>
```

Now we have specified all parameter and can run the function

```
> RF.estimate <- MCRestimate(golubTrain,
+ class.colum, classification.fun = "RF.wrap",
+ thePreprocessingMethods = Preprocessingfunctions,
+ poss.parameters = list.of.poss.parameter,
+ cross.outer = cross.outer, cross.inner = cross.inner,
+ cross.repeat = cross.repeat, plot.label = plot.label)</pre>
```

The result is an element of class MCRestimate

#### > RF.estimate

Result of MCRestimate with 3 repetitions of 2-fold cross-validation

Preprocessing function 1 : varSel.highest.var

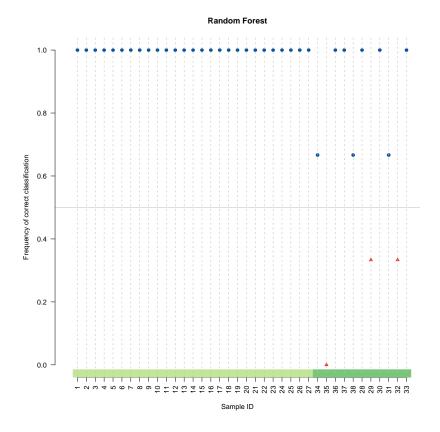
Classification function : RF.wrap

#### The confusion table:

ALL 27 0 0.000 AML 3 8 0.273

For each group we see how many samples have been misclassified most of the time. We can also visualize the result. The plot shows for each sample the number of times it has been classified correctly.

> plot(RF.estimate, rownames.from.object = TRUE,
+ main = "Random Forest")



## New data

We have estimated the misclassification rate of our complete algorithm. Because this seems to be a quit good result, we want to use this algorithm to classify new data. The function ClassifierBuild is used to build a classifier that can be used for new data.

```
> RF.classifier <- ClassifierBuild(golubTrain,
+ class.colum, classification.fun = "RF.wrap",
+ thePreprocessingMethods = Preprocessingfunctions,
+ poss.parameters = list.of.poss.parameter,
+ cross.inner = cross.inner)</pre>
```

The result of the function is a list with various arguments.

```
> names(RF.classifier)
```

```
[1] "classifier.for.matrix"
[2] "classifier.for.exprSet"
[3] "classes"
[4] "parameter"
[5] "thePreprocessingMethods"
[6] "class.method"
```

[7] "cross.inner"
[8] "information"

The most important elements of this list are classifier.for.matrix and classifier.for.exprSet. These can now be used to classify new data. The first one can be used if the new data is given by a matrix and the second will be used if the new data is given by a exprSet. Here we want to classify the data from the exprSet golubTest, that is also part of the package golubEsets.

This work was supported by NGFN.

# References

[1] Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA, Bloomfield CD, Lander ES. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring *Science* 286(5439): 531-7 (1999).