bootfs - Bootstrapped feature selection

Christian Bender *

September 8, 2013

This document describes the package 'bootfs' for robust feature selection in classification problems from high-throughput data, such as genomic or proteomic screening data. Several methods for classification are combined, in order to derive a robust estimation of the importance of each feature used for classification.

1 Using bootfs

This section contains the basic steps for analysis of high-throughput data, consisting of a number of samples to classify (such as patients) and a number of features, such as genes or proteins. Samples originate in two sample groups, for instance healthy versus sick patients. The aim is to find the most important features discriminating two or more classes within sample populations.

The usage of the package is illustrated for three classification algorithms: *pamr* (Prediction analysis for Microarrays, [7], implementation in *pamr*-R-package), *rf_boruta* (Random forests with the Boruta algorithm for feature selection, [4], implementation in *Boruta*-R-package) and *scad* (Support Vector Machines with Smoothly Clipped Absolute Deviation feature selection, [8], implementation in the *penalizedSVM* R-package [1]). Also available feature selection methods (through *penalizedSVM* package) are *1norm* for L1-penalisation (LASSO), *scad+L2* for Elastic-SCAD and *DrHSVM* for Elastic Net. Further, the original random forest implementation (*randomForest*-R-package [6]) is available as method *rf*, as well as the gradient boosting machine ([2, 3]) from package *gbm* as method *gbm*. The methods *pamr*, *rf*, *rf_boruta* and *gbm* also allow multi-class classification. First of all load the package:

*Translational Oncology (TRON) Mainz, Germany. eMail: christian.bender@tron-mainz.de

```
> library(bootfs)
```

1.1 Simulating data

Data can be simulated using a built in function *simDataSet*. This samples a data matrix with *nsam* samples and *ngen* genes, as well as a grouping vector defining two sample groups.

```
> set.seed(1234)
> data <- simDataSet(nsam=30, ngen=100, sigma=1.5, plot=TRUE)
> logX <- data$logX
> groupings <- data$groupings
```

1.2 Assessing performance of the classifiers

The first step is to verify, if the feature selection algorithms perform sufficienctly on the given data set. For this, a crossvalidation of different classification algorithms is run:

```
> ## run the crossvalidation
> ## note the number of repeats should be set to 10 or so,
> ## it is set to 2 here to have a low running time of this illustration
> ## create a parameter object used for the different methods
> # for crossvalidation
> paramsCV <- control_params(seed=123,
+  ncv=5, repeats=2, jitter=FALSE,      ## general parameters
+  maxiter=100, maxevals=50,             ## svm parameters
+  max_allowed_feat=500, n.threshold=50, ## pamr parameters
+  maxRuns=300,                          ## RF parameters
+  ntree = 1000,                         ## GBM parameters
+  shrinkage = 0.01, interaction.depth = 3,
+  bag.fraction = 0.75, train.fraction = 0.75,
+  n.minobsinnode = 3, n.cores = 1,
+  verbose = TRUE)
> ## run the crossvalidation
> ## takes a while
> methods <- c("pamr", "scad", "rf_boruta")
> retCV <- doCV(logX, groupings, fs.methods = methods, DIR = NULL, params=paramsCV)
```

The above command uses the classification methods PAMR, SCAD-SVM and RF-Boruta and performs a 5-fold (ncv=5) crossvalidation on the training data, repeating the crossvalidation 2 times with different training/test

set assignments. Classification methods and k-fold for the CV can be easily exchanged by setting the appropriate parameters. The results of the crossvalidation are summarised as ROC (Receiver operator characteristic) curves together with the corresponding AUC (area under the curve), shown in figures 1, 2 and 3.

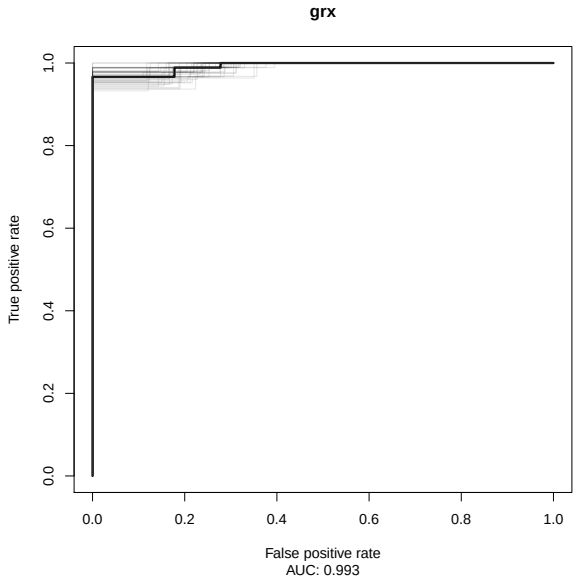

Figure 1: Receiver Operator Characteristic (ROC) curve for the PAM algorithm cross-validation. Input data were the simulated data matrix and grouping vectore for 30 samples and 100 features.

1.3 Do the feature selection and importance ranking

If the performance is of sufficient quality, as seen in the ROC curves generated during the cross-validation, the bootstrapping approach for deriving the most important features for this classification task can be done. Again, we select the three algorithms from above and perform bootstrapping on the input data. For each bootstrapping data set the feature selections are done using each algorithm.

```
> # for bootstrapping
> paramsBS <- control_params(seed=123,
+  jitter=FALSE, bstr=15,                  ## general parameters
+  maxiter=100, maxevals=50, bounds=NULL,  ## svm parameters
+  max_allowed_feat=500, n.threshold=50,   ## pamr parameters
+  maxRuns=300,                            ## RF parameters
```

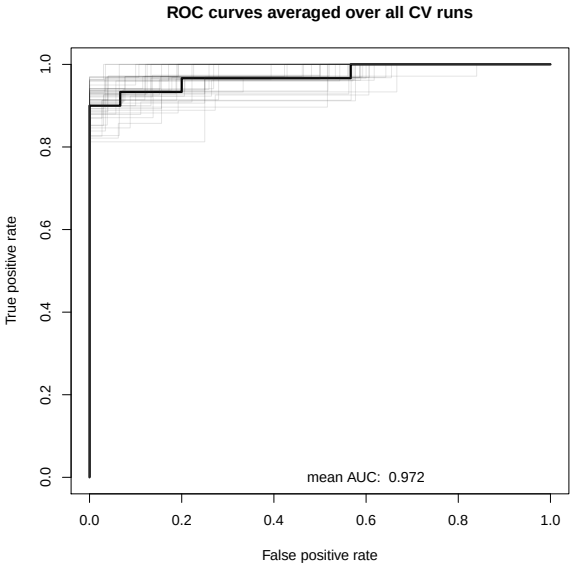

Figure 2: Receiver Operator Characteristic (ROC) curve for the random forest RF-Boruta algorithm cross-validation. Input data were the simulated data matrix and grouping vectore for 30 samples and 100 features.

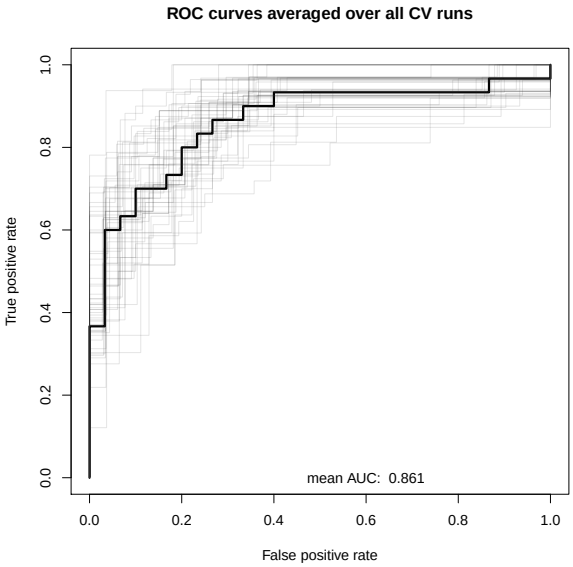

Figure 3: Receiver Operator Characteristic (ROC) curve for the SCAD-SVM algorithm cross-validation. Input data were the simulated data matrix and grouping vectore for 30 samples and 100 features.

```
+  ntree = 1000,                              ## GBM parameters
+  shrinkage = 0.01, interaction.depth = 3,
+  bag.fraction = 0.75, train.fraction = 0.75,
+  n.minobsinnode = 3, n.cores = 1,
+  verbose = TRUE, saveres=FALSE
+ )
> ## run the bootstrapping
> ## takes a while
> methods <- c("pamr", "scad", "rf_boruta")
> retBS <- doBS(logX, groupings, fs.methods=methods, DIR="bs", params=paramsBS)
>
```

Here, 15 bootstrap sample sets are drawn (bstr=15) and feature Selection is done for each sample set. The group proportions are kept constant during the sample selection. Now, for each method, a separate importance graph can be generated:

```
> ## show an importance ranking for a single
> ## classification method
> bsres <- makeIG(retBS[[1]], SUBDIR=NULL, prob=.999)
```

This might be useful to inspect the selected features for each method separately. The parameter *prob=.9* is used to define the cutoff value, how often a feature at least must co-occur with another feature, in which case an edge is drawn between them. However, the general ranking of the importance of the features is done by generating the combined importance graph from all selected methods:

```
> ## create the combined importance graph for all methods
> ## and export the adjacency matrix containing the
> ## numbers of occuerrences of the features, as well
> ## as the top hits.
> res <- resultBS(retBS, DIR=NULL, vlabel.cex = 3, filter = 5)
```

There are several arguments which customise the look of the importance graph. In this call, the *vlabel.cex* argument defines the magnification factor of the node labels (each node corresponds to one feature). The argument *filter* can be used to specify, how often a feature must co-occur with another, such that an edge is drawn between the two features. Figure 4 shows the result of the above call.

The graph can be customised more flexibly using the *importance_igraph* function directly:

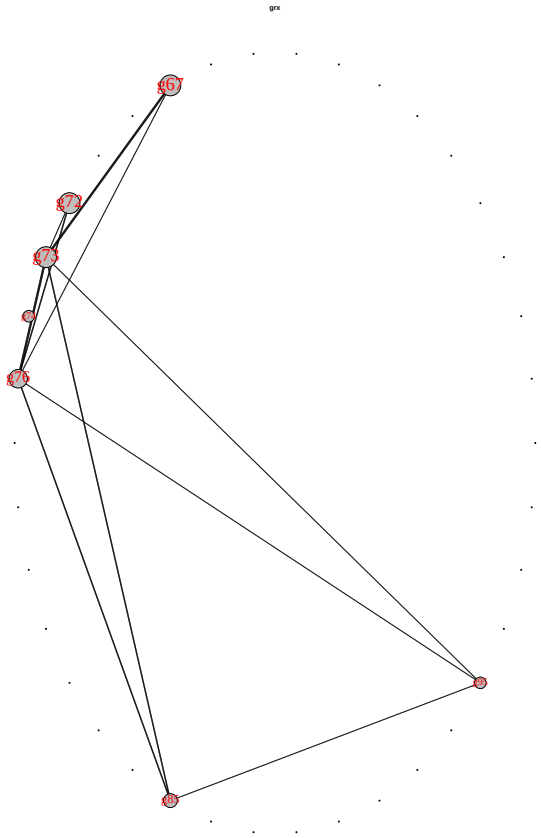

Figure 4: Importance graph generated by *resultBS*. Node width is proportional to the absolute frequency of occurrence of a feature across all bootstrap feature selections. Edge width is proportional to the frequency of co-occurrence of the two adjacent nodes.

```
> ## plot the importance graph directly. Gives more
> ## flexibility to adjust the graph
>
> resx <- res[[1]]
> ig <- importance_igraph(resx$adj, main = "my test",
+  highlight = NULL,        layout="layout.ellipsis",
+  pdf=NULL, pointsize=12, tk=FALSE,
+  node.color="grey", node.filter=NULL,
+  vlabel.cex=2, vlabel.cex.min=0.5, vlabel.cex.max=5,
+  max_node_cex=8,
+  edge.width=2, edge.filter=2, max_edge_cex=5, ewprop=3 )
```

Figure 5 shows the importance graph generated with the above call. Arguments *vlabel.cex*, *vlabel.cex.min* and *vlabel.cex.max* can be used to adjust the overall, minimum and maximum expansion factor for the node labels. *max_node_cex* controls the maximum expansion factor of the node size. The size of the nodes is always proportional to the absolute occurrence of the feature in the *bstr* bootstrapping sample sets. *edge.width* and *max_edge_cex* are used for setting the edge width and expansion factor for the edges, respectively, while *ewprop* is a proportionality factor controlling the decrease of edge width with decreasing frequency. A higher value of *ewprop* means a fast reduction of edge width and thus a less densly packed importance graph plot. Also consider the help pages for the respective functions, to learn about the remaining function arguments.

1.4 Doing multi-class classification

It is also possible to classify multiple classes at once. To illustrate this, we show an analysis of Fisher's Iris Data set. The goal is to discriminate between three Iris species (*Iris setosa*, *Iris virginica* and *Iris versicolor*), using four features *Sepal length, Sepal width, Petal length and Petal width.* We load the Iris data set and prepare it for bootfs:

```
> ## do multiclass classification
> ## load the data
> data(iris)
> groupings <- list(Species=iris$Species)
> logX <- iris[,1:4]
> methods <- c("gbm","rf","pamr")
> paramsCV <- control_params(seed=123,
+  ncv=5, repeats=2, jitter=FALSE,      ## general parameters
+  maxiter=100, maxevals=50,             ## svm parameters
```

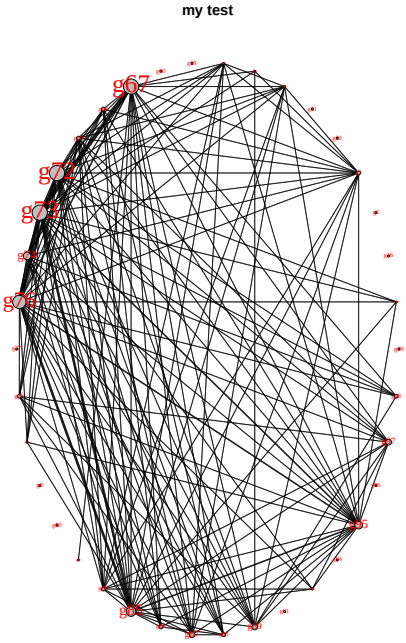

Figure 5: Importance graph generated by call to *importance_igraph* directly. Note how the edge widths are thicker but decreasing more rapidly as in figure 4, achieved by setting *max_edge_cex* and *ewprop* appropriately. Besides, the node label sizes are adjusted using *vlabel.cex*.

```
+  max_allowed_feat=500, n.threshold=50, ## pamr parameters
+  maxRuns=300,                          ## RF parameters
+  ntree = 1000,                         ## GBM parameters
+  shrinkage = 0.01, interaction.depth = 3,
+  bag.fraction = 0.75, train.fraction = 0.75,
+  n.minobsinnode = 3, n.cores = 1,
+  verbose = TRUE)
```

Here we use the Gradient Boosting Machine, Breiman's Random Forests and PAMR. As usual we perform a crossvalidation of the selected methods, to assess their individual performance:

```
> ## crossvalidation
> retCV <- doCV(logX, groupings, fs.methods = methods, DIR = NULL, params=paramsCV)
```

A summary of the CV results can be obtained using the *resultsCV* function:

```
> resultCV(retCV)
```

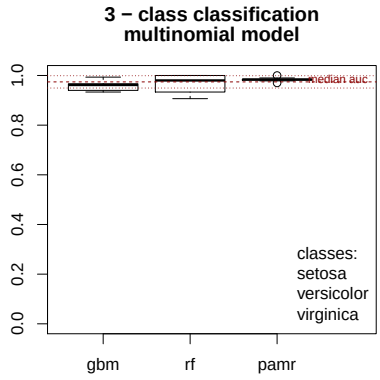

Figure 6: Multiclass AUCs generated with the *pROC* package. Shown are the distributions of AUCs over all cross-validation repeats and folds.

Note that ROC curve display is not possible for more than two classes. For assessment of the performance we rely on the area under curve calculation for multiclass problems [5], implemented in the *pROC* package. this gives us the AUC for each CV fold and repeat.

Now, we can proceed to the bootstrapped feature selection:

```
> paramsBS <- control_params(seed=123,
+  jitter=FALSE, bstr=15,                  ## general parameters
+  maxiter=100, maxevals=50, bounds=NULL,  ## svm parameters
+  max_allowed_feat=500, n.threshold=50,   ## pamr parameters
+  maxRuns=300,                            ## RF parameters
+  ntree = 1000,                           ## GBM parameters
+  shrinkage = 0.01, interaction.depth = 3,
+  bag.fraction = 0.75, train.fraction = 0.75,
+  n.minobsinnode = 3, n.cores = 1,
+  verbose = TRUE, saveres=FALSE
+ )
> ## bootstrapped feature selection
> retBS <- doBS(logX, groupings, fs.methods=methods, DIR=NULL, params=paramsBS)
```

The results are generated and the importance graph is plotted:

```
> ## make results from all methods used
> res <- resultBS(retBS, DIR=NULL, vlabel.cex = 3, filter = 1)
> ## plot the importance graph
> resx <- res[[1]]
> ig <- importance_igraph(resx$adj, main = "multiclass test, IRIS data",
+         highlight = NULL,        layout="layout.ellipsis",
+                 pdf=NULL, pointsize=12, tk=FALSE,
+                 node.color="grey", node.filter=NULL,
+                 vlabel.cex=1.2, vlabel.cex.min=0.5, vlabel.cex.max=4,
+                 max_node_cex=8,
+         edge.width=1, edge.filter=1, max_edge_cex=2, ewprop=3 )
```

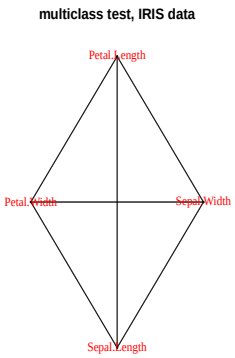

Figure 7: Result of multiclass bootfs, done on the IRIS data set. All 4 features are left.

Session Information

The version number of R and packages loaded for generating the vignette were:

- R version 3.0.1 (2013-05-16), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=en_US.UTF-8`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=C`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: BiocInstaller˜1.10.3
- Loaded via a namespace (and not attached): tools˜3.0.1

References

[1] Natalia Becker, Grischa Toedt, Peter Lichter, and Axel Benner. Elastic scad as a novel penalization method for svm classification tasks in high-dimensional data. *BMC Bioinformatics*, 12:138, 2011.

[2] Jerome˜H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[3] Jerome˜H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[4] Miron˜B. Kursa and Witold˜R. Rudnicki. Feature selection with the boruta package. *Journal of Statistical Software*, 36(11):1–13, 9 2010.

[5] Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez, and Markus Müller. proc: an open-source package for r and s+ to analyze and compare roc curves. *BMC bioinformatics*, 12(1):77, 2011.

[6] Leo˜Breiman Statistics and Leo Breiman. Random forests. In *Machine Learning*, pages 5–32, 2001.

[7] Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proc Natl Acad Sci U S A*, 99(10):6567–6572, May 2002.

[8] Hao˜Helen Zhang, Jeongyoun Ahn, Xiaodong Lin, and Cheolwoo Park. Gene selection using support vector machines with non-convex penalty. *Bioinformatics*, 22(1):88–95, Jan 2006. SCAD paper.