Stepwise: Stepwise Classification of Cancer Samples using Clinical and Molecular Data

Askar Obulkasim

Department of Epidemiology and Biostatistics, VU University Medical Center
P.O. Box 7075, 1007 MB Amsterdam, The Netherlands
askar.wubulikasimu@vumc.nl

Contents

1 Introduction

This vignette shows the use of the `Stepwise` package for the integrative analysis of clinical risk factor and molecular data. For more detailed information on the algorithm and assumptions we refer to the article (Obulkasim *et al*, 2011) and its supplementary material. The following features are discussed in detail:

- Prediction with user defined algorithm.
- Proximity matrix calculation.
- Reclassification score (RS) calculation.
- Reference curve generation.
- Assigning test samples to one of the two data types.

These are illustrated on an example data set, which is introduced first.

2 Central Nervous System cancer data (CNS)

The CNS tumor data (Pomeroy *et al.*, 2002) has been used to predict the response of childhood malignant embryonal tumors of CNS to the therapy. The data set is composed of 60 patients (biopsies were obtained before they received any treatment), 21 patients died and 39 survived within 24 months. Gene expression data has 7128 genes and clinical features are Chang stage (nominal), sex (binary), age (nominal), chemo Cx (binary), chemo VP (binary).
Data can be downloaded as an .RData file from `http://www.math.univ-toulouse.fr/ lecao/package.html`.

```
> # load the full Central Nervous System cancer data
> library(stepwiseCM)
> data(CNS)
```

The code above loads a list object `CNS` with following components

- `mrna:` A matrix containing the expression values of 7128 genes (rows) in 60 samples (columns).
- `cli:` A matrix containing the five clinical risk factors: ChangStage, Sex, Age, chemo Cx, chemo VP.
- `class:` A vector of the class labels. Died patients are denoted with 0 and survived patients are denoted with 1.
- `gene.name` A data frame containing the gene names and descriptions.

3 Prediction

The first step of an integrative analysis comprises obtaining the prediction labels of set training set using the two data types independently. The objective of this prediction step is to assess the classification power of the two data sets. The prediction labels of the train set will be attained by using the chosen classification algorithm. One would specify the type of algorithms via the `type` parameter. Currently, nine methods are implemented. Note that TSP, PAM, plsrf_x and plsrf_x_pv algorithms do not work with clinical data. We can attain the prediction labels of the training set either with the Leave-One-Out-Cross-Validation (LOOCV) or the K-fold cross validations by specifying the `CVtype` option. If set to the `k-fold`, then we have to specify the value of `outerkfold` (the number of folds). If set to the `loocv`, algorithm ignore the `outerkfold` parameter value and the `loocv` will be performed. The parameter `innerkfold` defines the number of cross validation used to estimate the model parameter(s) e.g. λ in the `GLM_L1` (Goeman, 2011). If one also want to extract the selected features names during the training and the test phase, the parameter `featurenames` should be specified.
Here, we choose to use the `RF` option for clinical data and `plsrf_x` (Boulesteix *et al*, 2008) option for expression data, respectively. For computational convenience we set `CVtype="k-fold"`, `outerkfold=2` and `innerkfold=2`. The optimal parameter `test` is also given.

```
> # gain the prediction labels of the training set
> set.seed(123L)
> data(CNS)
> attach(CNS)
> #Note that columns of the train set should corresponding the sample and rows
> #corresponding the feature.
> train.exp <- mrna[, 1:30]
```

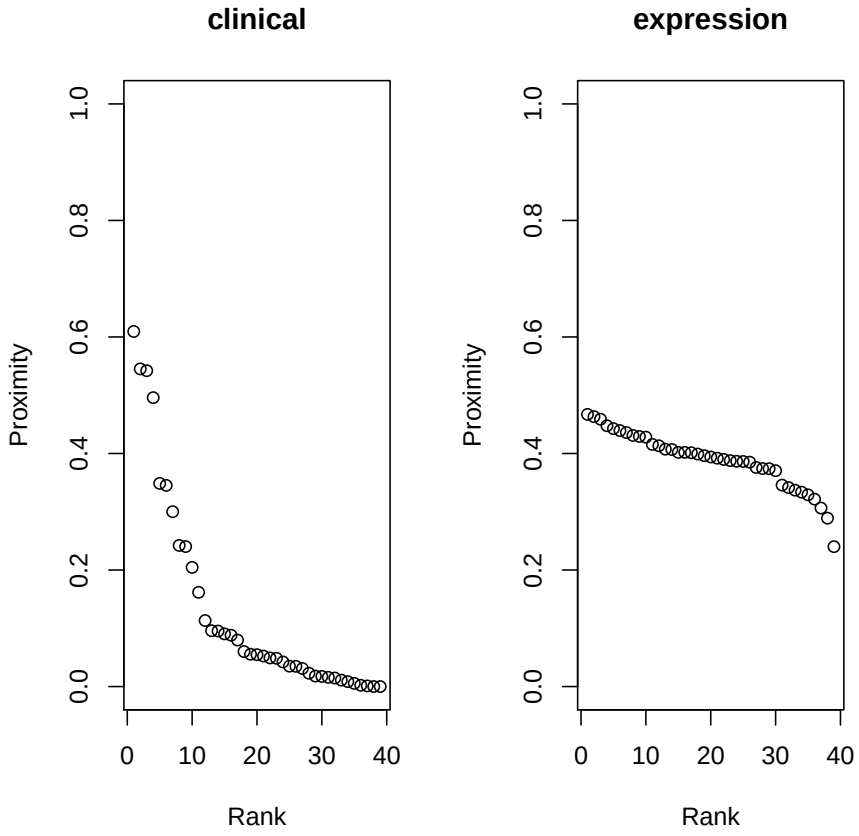
```
> train.cli <- t(cli[1:30, ])
> train.label <- class[1:30]
> test.exp <- mrna[, 31:60]
> test.cli <- t(cli[31:60, ])
> pred.exp <- Classifier(train=train.cli, test=test.cli, train.label=train.label,
+                        type="RF", CVtype="k-fold", outerkfold=2, innerkfold=2)
> pred.cli <- Classifier(train=train.exp, test=test.exp, train.label=train.label,
+                        type="plsrf_x", CVtype="k-fold", outerkfold=2, innerkfold=2)
> # Classification accuracy of the training set from clinical data is:
> sum(pred.cli$P.train == train.label)/length(train.label)
> # Classification Accuracy of the training set from expression data is:
> sum(pred.exp$P.train == train.label)/length(train.label)
```

We can also obtain the prediction labels using the `Classifier.par` function if the multiple cores computation is desired. If this function is used, one can take advantage of its powerful parallel computation capability by specifying the `ncpus` option which controls how many CPUs are assign to the computation.

4 The proximity matrix calculation

The second step is to calculate the proximity matrix of the training set with the random forest algorithm (Breiman , 2001). This matrix is used to locate the test sample's position in the clinical data space with available clinical information and locate the "pseudo" position of the test sample in the expression data space by the indirect mapping (IM). As in the prediction step, we have to calculate two proximity matrices using the two data independently. Because of the large number of irrelevant feature in the expression data, the value of `N` need to be set high, so that final averaged proximity matrix will cancel out the irrelevant features effect. Here, we set `N=2` and calculation is done without the parallel processing procedure.

```
> train.exp <- mrna[, 1:40]
> train.cli <- t(cli[1:40, ])
> train.label <- class[1:40]
> test.cli <- t(cli[41:60, ])
> prox <- Proximity(train.cli=train.cli, train.label=train.label, test.cli=test.cli,
+                   train.gen=train.exp, N = 2)
> prox1 <- Proximity(train.cli=train.cli, train.label=train.label, test.cli=test.cli,
+                   train.gen=train.cli, N = 2)
> #check the range of proximity values from two different data types
> par(mfrow=c(1,2))
> plot(sort(prox1$Prox.gen[1, ][-1], decreasing=TRUE), main="clinical", xlab="Rank",
+      ylab="Proximity", ylim=c(0,1))
> plot(sort(prox$Prox.gen[1, ][-1], decreasing=TRUE), main="expression", xlab="Rank",
+      ylab="Proximity", ylim=c(0,1))
```



The range of proximity values from two different data types can be visualized in following way: choose the first sample in the training set and order (descending) the rest of the samples proximity values (from clinical data) w.r.t this sample. Apply similar procedure to the proximity matrix from the expression data. As Figure shows, two types of data produce the proximity values with different ranges. Large range is observed in the clinical data setting (0-0.6), small range is observed in the expression data setting (0.5-0.2). Note, since the tree construction in side the RF has randomness, the resulting proximity matrix is may not be the same in each run (but the different will not be too big).

5 The reclassification score (RS) calculation

After finishing the first two steps, we received all the inputs requires for the calculation of the RS. The RS can be attained by using the `RS.generator` function. For this function, we only need to specify which type of similarity measure is should be used during the calculation. If `type = "proximity"`, then RS is calculated directly from the proximity values. If `type = "rank"`, function first order the proximity values and uses their ranking in stead. Latter one is more robust against the different ranges problem as we observed previously. Here, in order see the differences in the outputs from two methods, we set `type = "both"`

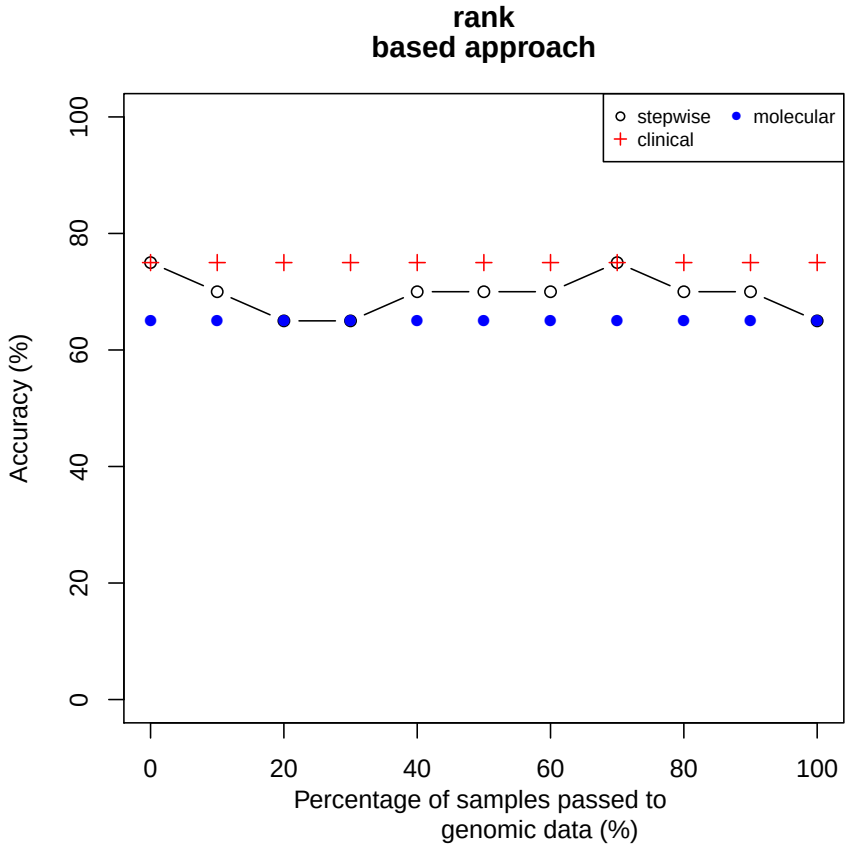
```
> train.exp <- mrna[, 1:30]
> train.cli <- t(cli[1:30, ])
```

```
> train.label <- class[1:30]
> test.cli <- t(cli[31:60, ])
> pred.cli <- Classifier(train=train.cli, test=c(), train.label=train.label, type="RF",
+                        CVtype="k-fold", outerkfold=2, innerkfold=2)
> pred.exp <- Classifier(train=train.exp, test=c(), train.label=train.label,type="GLM_L1",
+                        CVtype="k-fold", outerkfold=2, innerkfold=2)
> prox <- Proximity(train.cli=train.cli, train.label=train.label, test.cli=test.cli,
+                   train.gen=train.exp, N = 2, Parallel = FALSE)
> RS <- RS.generator(pred.cli=pred.cli$P.train, pred.gen=pred.exp$P.train,
+        train.label=train.label, prox.gen=prox$Prox.gen, prox.cli=prox$Prox.cli,
+        type = "both")
> #observe the differences by ranking the RS values
> order(RS[, 1]) # from the ranking approach
> order(RS[, 2]) # from the proximity approach
```

6 Reference curve generation

To choose a RS threshold which produces reasonably high accuracy and at the same time keep large proportion samples at the first stage, we need a reference curve which shows how accuracy changes when different percentage of samples are classified with molecular data. For this, we need a training set samples and a data matrix which is not used in previous step.

```
> tr.exp <- mrna[, 1:40]
> tr.cli <- t(cli[1:40, ])
> tr.label <- class[1:40]
> te.exp <- mrna[, 41:60]
> te.cli <- t(cli[41:60, ])
> te.label <- class[41:60]
> result <- Curve.generator(train.cli=tr.cli, train.gen=tr.exp, train.label=tr.label,
+                test.cli= te.cli, test.gen=te.exp, test.label=te.label,
+                type=c("RF", "TSP"), RStype = "rank", Parallel = FALSE,
+                CVtype = "k-fold", outerkfold = 2, innerkfold = 2,
+                N = 2, plot.it=TRUE)
>
```



Accuracy plot can be generated by setting "plot.it=TRUE". In this plot X axis denotes the percentage of samples classified with expression data and Y accuracy denotes the corresponding accuracy at that point. Note that since the tree construction inside the RF has randomness, so the resulting proximity matrix may not be the same in each run. This leads to different RS results in each time (but the difference will not be too big).

7 Assigning test samples to one of the two data types

Once we obtain the reference curve utilizing the available samples for which both clinical and molecular information are available, now we choose a RS threshold which pass specified percentage of samples to molecular data. We can use this threshold hold to all the incoming samples to decide whether we should generate its molecular profile or simply classify it using clinical information.

```
> tr.cli <- t(cli[1:40, ])
> te.cli <- t(cli[41:60, ])
> tr.gen <- mrna[, 1:40]
> te.gen <- mrna[, 41:60]
> tr.label <- class[1:40]
> te.label <- class[41:60]
> result <- Curve.generator(train.cli=tr.cli, train.gen=tr.exp, train.label=tr.label,
+                test.cli= te.cli, test.gen=te.exp, test.label=te.label,
```

```
+                 type=c("RF", "plsrf_x"), RStype = "rank", Parallel = FALSE,
+                 CVtype = "k-fold", outerkfold = 2, innerkfold = 2,
+                 N = 2, plot.it = FALSE)
> A <- Step.pred(result, test.cli, 30)
```

Here we allow 30% of samples can be classified with expression data. This function gives us the predicted class labels obtained by using the clinical information, the RS threshold which corresponding to the 30% re-classification and a vector of binary values indicating which samples are recommended to be classified with molecular data.

References

Obulkasim,A., Meijer, A.G. and van de Wiel, A.M. (2011). Stepwise Classification of Cancer Samples using Clinical and Molecular Data. *BMC Bioinformatics*, in press.

Breiman, L., (2001). Random Forests. *Machine Learning*, **45**, 5–32.

Pomeroy, S. L., Tamayo, P. and Gaasenbeek, M. (2002). Prediction of Central Nervous System Embryonal Tumour Outcome Based on Gene Expression. *Nature*, **415**, 436–442.

Boulesteix,A-L., Porzelius,C. and Daumer,M. (2008). Microarray-based Classification and Clinical Predictors: on Combined Classifiers and Additional Predictive Value. *Bioinformatics*, **24**, 1698-1706.