Bioconductor's maPredictDSC package

Adi L. Tarca[1,2,3]

October 17, 2016

[1]Department of Computer Science, Wayne State University
[2]Bioinformatics and Computational Biology Unit of the NIH Perinatology Research Branch
[3]Center for Molecular Medicine and Genetics, Wayne State University

1 Overview

This package implements the classification pipeline of the best overall team (Team221) (see **?**) in the IMPROVER Diagnostic Signature Challenge described in **?**. Additional capability is added to explore other combinations of methods for data preprocessing, feature ranking and classification described in **?**. In a nutshell, with this package one starts with Affymetrix .CEL expression files (all platforms supported) some of which correspond to a set of training samples (class is required, 2 classes only) while some other correspond to test samples for which the class will be predicted. One or more models are built on the training data, and predictions are made on the test samples. Several performance metrics used in the IMPROVER DSC can be computed for the fitted models if the class of the test samples is known including the Area Under the Precision-Recall Curve (AUPR), Belief Confusion Metric (BCM) and Correct Class Enrichment Metric (CCEM). Note that the sample size for this example as well as the arguments in the function calls below were chosen to limit the amount of time required to run the example on a decent computer (max 5 mins, as required by the Bioconductor standards). See the cited references for results on several datasets of much larger sample size and more appropriate values for the arguments in the function calls.

2 Developing prediction models with maPredictDSC package

This document provides basic introduction on how to use the `maPredictDSC` package. For extended description of the methods used in this package please consult these references: **?** and **?**.

We demonstrate the functionality of this package using a set of lung cancer samples obtained using Affymetrix HG-U133 Plus 2.0 technology that are available from GEO. In this example we use 7 Adenocarcinoma (AC) and 8 Squamous cell carcinoma (SCC) samples taken at random from 3 GEO datasets (GSE10245, GSE18842 and GSE2109) and 15 samples used for testing purpose from a dataset produced by the organizers of the IMPROVER Diagnostic Signature Challenge also available from GEO (GSE43580). The data is available in the `LungCancerACvsSCCGEO` package. The assignment of the samples into groups is defined in the `anoLC` data frame available by loading the `LungCancerACvsSCCGEO` datset as shown below:

```
> library(maPredictDSC)
> library(LungCancerACvsSCCGEO)
> data(LungCancerACvsSCCGEO)
> anoLC
```

	files	group
1	GSM137916.CEL.gz	AC
2	GSM258560.CEL.gz	AC
3	GSM258579.CEL.gz	AC
4	GSM258589.CEL.gz	AC
5	GSM258598.CEL.gz	AC
6	GSM353885.CEL.gz	AC
7	GSM467021.CEL.gz	AC
8	GSM152624.CEL.gz	SCC
9	GSM258580.CEL.gz	SCC
10	GSM277678.CEL.gz	SCC
11	GSM466956.CEL.gz	SCC
12	GSM466980.CEL.gz	SCC
13	GSM466989.CEL.gz	SCC
14	GSM467023.CEL.gz	SCC
15	GSM46976.CEL.gz	SCC
16	lung_100.CEL	Test
17	lung_107.CEL	Test
18	lung_111.CEL	Test
19	lung_15.CEL	Test
20	lung_150.CEL	Test
21	lung_29.CEL	Test
22	lung_30.CEL	Test
23	lung_35.CEL	Test
24	lung_40.CEL	Test
25	lung_41.CEL	Test
26	lung_50.CEL	Test
27	lung_51.CEL	Test
28	lung_59.CEL	Test
29	lung_62.CEL	Test
30	lung_8.CEL	Test

```
> gsLC
```

	SCC	AC
lung_15.CEL	1	0
lung_29.CEL	0	1
lung_30.CEL	0	1
lung_59.CEL	0	1
lung_51.CEL	1	0
lung_100.CEL	1	0

```
lung_62.CEL    1  0
lung_8.CEL     1  0
lung_41.CEL    0  1
lung_40.CEL    1  0
lung_150.CEL   1  0
lung_111.CEL   1  0
lung_35.CEL    0  1
lung_107.CEL   1  0
lung_50.CEL    0  1
```

The data frame `gsLC` included also in this dataset gives the class of the test samples that we will use later to assess the predictions of different models produced by the `predictDSC` function which is the main function of the package. The `predictDSC` function takes as input a folder of raw Affymetrix CEL files and explores a set of combinations of data preprocessing (rma, gcrma, mas5), feature ranking methods (t-test, moderated t-test, wilcoxon test) and classifier types (LDA, SVM, kNN). For each such combination, the optimal number of genes to be used in the model is automatically determined by optimizing the AUC statistic computed via cross-validation on the training data. Also, for each combination, a final model is fitted using all training data, and predictions on the "Test" samples (defined as such in the `ano` data frame) are computed.

```
> modlist=predictDSC(ano=anoLC,
+ celfile.path=system.file("extdata/lungcancer",package="LungCancerACvsSCCGEO"),
+ annotation="hgu133plus2.db", preprocs=c("rma"),
+ filters=c("mttest","ttest"),classifiers=c("LDA","kNN"),
+ CVP=2,NF=4, NR=1,FCT=1.0)

Background correcting
Normalizing
Calculating Expression
Getting probe level data...
Computing p-values
Making P/M/A Calls
rma_mttest_LDA
rma_ttest_LDA
rma_mttest_kNN
rma_ttest_kNN
```

In addition to the 27 models that can be fitted with the simple call of the function above, one can obtain 27 additional models by changing the FCT (fold change threshold) from 1.0 to say 1.25 or 1.5 fold. This will exclude genes from being potential candidate to be included in the model if the change in expression on the current training data fold is not above FCT. Note, if there are not at least NF features meeting the fold change required threshold, the threshold will be ignored and features will be selected from the top ones sorted by p-values.
We can explore the details recorded for each methods combination stored in the elements of modlist:

```
> modlist[["rma_ttest_LDA"]]
```

```
$predictions
                SCC     AC
lung_100.CEL 0.0006 0.9994
lung_107.CEL 1.0000 0.0000
lung_111.CEL 0.9990 0.0010
lung_15.CEL  1.0000 0.0000
lung_150.CEL 1.0000 0.0000
lung_29.CEL  0.0000 1.0000
lung_30.CEL  0.0000 1.0000
lung_35.CEL  0.0000 1.0000
lung_40.CEL  1.0000 0.0000
lung_41.CEL  0.0000 1.0000
lung_50.CEL  0.0001 0.9999
lung_51.CEL  1.0000 0.0000
lung_59.CEL  0.0000 1.0000
lung_62.CEL  1.0000 0.0000
lung_8.CEL   1.0000 0.0000

$features
[1] "F225214_at"   "F212900_at"   "F204703_at"   "F202973_x_at"

$model
Call:
lda(formula(paste("CLS~1", vr, sep = "+")), data = mydat, prior = c(1,
    1)/2)

Prior probabilities of groups:
  0   1
0.5 0.5

Group means:
  F225214_at F212900_at F204703_at F202973_x_at
0   8.048657   8.782127   7.823691    10.068043
1   7.137370   7.878274   7.086235     8.503033

Coefficients of linear discriminants:
                   LD1
F225214_at   -2.5547801
F212900_at   -2.7501437
F204703_at   -3.9426705
F202973_x_at -0.2958261

$performanceTr
     NN   meanAUC     sdAUC
[1,]  2 0.5208333 0.2062395
```

```
[2,]  3 0.7343750 0.3756505
[3,]  4 0.8854167 0.1031197

$best_AUC
[1] 0.8854167
```

Note that the names of the features selected for this model which correspond to Affymetrix probesets have an "F" suffix added to their names since LDA does not like variable neames to start with a number.
The different combinations of methods can be ranked using the cross-validated AUC on the training data using:

```
> trainingAUC=sort(unlist(lapply(modlist,"[[","best_AUC")),decreasing=TRUE)
> cbind(trainingAUC)

               trainingAUC
rma_mttest_LDA   0.9843750
rma_mttest_kNN   0.9375000
rma_ttest_LDA    0.8854167
rma_ttest_kNN    0.8750000
```

Now the model that apears to be best using the AUC on the training data will not necessarily be best according to the same or other statistics on the test data. To illustrate this, we will compute various metrics such as BCM, CCEM and AUPR implemented in the `perfDSC` function for these models on the test data:

```
> perF=function(out){
+ perfDSC(pred=out$predictions,gs=gsLC)
+ }
> testPerf=t(data.frame(lapply(modlist,perF)))
> testPerf=testPerf[order(testPerf[,"AUC"],decreasing=TRUE),]
> testPerf

                     BCM      CCEM      AUPR       AUC
rma_ttest_LDA  0.9444139 0.9333167 1.0000000 1.0000000
rma_mttest_LDA 0.8888778 0.8666600 0.9555556 0.8888889
rma_mttest_kNN 0.8888889 0.8666667 0.9555556 0.8888889
rma_ttest_kNN  0.8611111 0.8666667 0.8350309 0.8611111
```

We can also combine the predictions from several models aka "wisdom of crowds" by using the `aggregateDSC` function:

```
> best3=names(trainingAUC)[1:3]
> aggpred=aggregateDSC(modlist[best3])
> #test the aggregated model on the test data
> perfDSC(aggpred,gsLC)
```

```
      BCM       CCEM       AUPR        AUC
0.9073935 0.8777700 1.0000000 1.0000000
```

In this example combining the predictions from the best 3 models (as apparent on the training data) leads to better prediction on the test data compared to using the single best model chosen according to the training performance.