Tutorial for R package methylClass

Yu Liu

12/2/2021

Introduction

DNA methylation profiling is emerging as a useful tool to increase the accuracy of cancer diagnosis *[1, 2]*. However, a comprehensive R package specially for it is still lacking. Hence, we developed the R package *methylClass* to handle the issues about methylation-based classification. Within it, we provide the eSVM (ensemble-based support vector machine) model able to achieve a better accuracy in methylation data classification than the popular random forest (RF) model, and also overcomes the time-consuming problem of traditional SVM. In addition, some novel feature selection methods, such as *SCMER*, are included in the package to improve the classification *[3]*. Furthermore, in view that methylation data can be converted to other types of omics, such as copy number variation (CNV) data, we also provide several functions for multi-omics study.

Package installation

Code of *methylClass* is freely available at https://github.com/yuabrahamliu/methylClass.

The following commands can be used to install this R package.

```
library(devtools)

install_github('yuabrahamliu/methylClass')
```

Data preparation

To demonstrate the functions of *methylClass*, this tutorial uses a dataset that accompany with the package. It includes a matrix with 381 rows and 12000 columns. Each row represents a sample and each column represents a DNA methylation probe from Infinium EPIC BeadChip platform. The 12000 columns here are the top 12000 most variable probes in the dataset, and the values recorded in this matrix are the DNA methylation beta values detected from the samples, which are brain tumor tissues from donors. Among the 381 samples, 161 are GBM (glioblastoma) samples, 85 are IDH_Glioma samples, 72 are EPN (ependymona) samples, 21 are MNG (meningioma) samples, 19 are MB (medulloblastoma) samples, 12 are PA (pilocytic astrocytoma) samples, and 11 are PXA (pleomorphic xanthoastrocytoma) samples. The row names of the matrix are the sample IDs while the column names are the DNA methylation probe IDs.

In addition to this matrix, a factor is also with this dataset, which records the class labels for the 381 samples. Each element is a label for one sample and the element order is the same as the sample order in the beta value matrix.

Now, attach *methylClass* to the R session and get the example data.

```
library(methylClass)

betas <- system.file('extdata', 'testbetas.rds', package = 'methylClass')
betas <- readRDS(betas)

labels <- system.file('extdata', 'testlabels.rds', package = 'methylClass')
labels <- readRDS(labels)
```

The beginning parts of the beta value matrix and the labels are shown below.

```
betas[1:6, 1:6]
#>       cg17386213 cg00450784 cg26483229 cg25834419 cg15650509 cg01663603
#> S728 0.417431193 0.06888500 0.02767962 0.08226950 0.01270772 0.03160920
#> S852 0.028862479 0.03899721 0.02587694 0.12622549 0.02272727 0.05666667
#> S918 0.976095618 0.97063291 0.96574344 0.92402659 0.92163009 0.97523220
#> T481 0.003842124 0.02400132 0.93372674 0.04285327 0.94285353 0.03452562
#> T834 0.080890052 0.60532931 0.17549443 0.23743340 0.17431657 0.09886629
#> T900 0.008695652 0.07142857 0.19154229 0.08108108 0.03313253 0.03357314
```

```
head(labels)
#> [1] MNG        MNG        IDH_Glioma EPN        GBM        GBM
#> Levels: GBM < IDH_Glioma < EPN < MNG < MB < PA < PXA
```

Cross validation

We will use these data to train 2 classifiers to distinguish different sample classes, one is an RF (random forest) classifier, the other is an eSVM (ensemble-based support vector machine) classifier, and we use a normal 5 fold cross validation (CV) method to evaluate their performances. This can be done via the functions `maincv` and `maincalibration` in the package.

We first use `maincv` to train the raw models in a 5 fold CV, and then use `maincalibration` to calibrate the raw model results and improve the performance.

For the RF CV models, we provide the 381 by 10000 betas matrix to `maincv` via its parameter `betas..` and transfer the labels via the paramter `y..`. We set another parameter `subset.CpGs` as 5000, so that for each CV loop, the top 5000 most variable probes will be selected and train the RF model for this loop. The default values of the parameters `n.cv.folds` is 5 and `normalcv` is FALSE, so that a 5 by 5 nested CV will be performed, but here we change `normalcv` to TRUE to save time, so that a normal 5 fold CV will be performed with only the 5 outer loops of the 5 by 5 nested CV. Then, we set `out.path` as "RFCV", so that a folder named "RFCV" will be created in the current working directory to save the result files of this function, and another parameter `out.fname` has a default value as "CVfold", so that all the files saved will have a prefix "CVfold" in their names. Finally, set the parameter `method` as "RF", so that RF models will be trained.

```
maincv(y.. = labels,
       betas.. = betas,
       subset.CpGs = 10000,
       n.cv.folds = 5,
       normalcv = TRUE,
       out.path = 'RFCV',
       out.fname = 'CVfold',
       method = 'RF',
       seed = 1234)
```

After this raw model training, we use the function `maincalibration` to calibrate the raw results. We transfer the `labels` to this function also via the parameter `y..`, and because `maincalibration` depends on the results of `maincv`, the folder with the `maincv` result files should be transferred to it via the parameter `load.path`, and the file name prefix should be transferred via `load.fname`. Actually, `load.path` and `load.fname` here match `out.path` and `out.fname` in `maincv`. Then, because `maincv` trains the models on normal CV loops, not the nested ones, here the `normalcv` parameter also need to be set as TRUE. Finally, tell the function the models are RF models via set the parameter `algorithm` as "RF".

```
RFres <- maincalibration(y.. = labels,
                         load.path = 'RFCV',
                         load.fname = 'CVfold',
                         normalcv = TRUE,
                         algorithm = 'RF',
                         setseed = 1234)
```

Then, the result `RFres` is a matrix with the performance of RF model in the 5 fold CV, for the raw model and all the calibrated models.

```
RFres
#>          misc.error auc.HandTill brier      mlogloss
#> rf       0.0656168  0.9937746    0.1986897  0.4688998
#> rf_LR    0.05511811 0.9827189    0.08251631 0.3270816
#> rf_FLR   0.06036745 0.9924814    0.0848257  0.191699
#> rf_MR    0.06036745 0.9933325    0.09845761 0.2331356
```

You can see the misclassification error of the raw RF model (rf) is around 0.0656, while all the 3 calibration methods, i.e., LR (logistic regression), FLR (Firth's regression) and MR (ridge regression), can reduce this error and the best results are from LR, it has an error of 0.0551.

The same things can be done to construct eSVM models for the 5 fold CV. Just note to change the parameters `method` and `algorithm` to "eSVM", and also set another folder name to save the `maincv` results for eSVM.

```
maincv(y.. = labels,
       betas.. = betas,
       subset.CpGs = 10000,
       n.cv.folds = 5,
       normalcv = TRUE,
       out.path = 'eSVMCV',
       out.fname = 'CVfold',
       method = 'eSVM',
       seed = 1234)

eSVMres <- maincalibration(y.. = labels,
                           load.path = 'eSVMCV',
                           load.fname = 'CVfold',
                           normalcv = TRUE,
                           algorithm = 'eSVM',
                           setseed = 1234)
```

We can see the result of eSVM via the returned `eSVMres`.

```
eSVMres
#>            misc.error auc.HandTill brier      mlogloss
#> esvm     0.05249344 0.9927924    0.09054083 0.2242651
#> esvm_LR  0.04724409 0.9848411    0.07675574 0.2597599
#> esvm_FLR 0.04986877 0.9867097    0.07542102 0.1824998
#> esvm_MR  0.05249344 0.9863797    0.0904642  0.2429746
```

Here, we use the beta values in the data matrix to train the models, and for methylation data, it is also recommended to convert these betas to M values and try to train models on them to see if there is any large difference between the model performance of beta values and M values.

For the beta value model here, we evaluate the performance using the 5 fold CV, but actually, the parameter `normalcv` can also be set as FALSE, and then a 5 by 5 nested CV will be used to evaluate the model. It will take more time than the normal 5 fold CV, but it can prevent the problem of over-fitting for the calibration step more effectively.

Another parameter need to be noted is `subset.CpG`. Here we set it as 10000, so that the top 10000 most variable probes are used to train the models, and any numeric number transferred to it will be explained as the number of top variable probes need to construct the model. While another 2 types of features can also be selected via set `subset.CpG` as the string "SCMER" or "limma", so that the method *SCMER* will be used to select the features able to preserve the manifold structure of the original data *[3]*, while *limma* will be used to select the top differential probes between the sample classes *[4]*. Then, these probes can be used to train the models, instead of the top variable ones. More details can be obtained from the help documents of the function.

For the RF and eSVM models trained here, if more details are needed for each CV loop, such as the testing sample prediction score matrix, the trained model object, etc, they can be found at the saved files generated by `maincv` and `maincalibration`. If don't need to know so much details, these files can be deleted after getting the final evaluation matrices of `RFres` and `eSVMres`.

Model construction

From `RFres` and `eSVMres`, we find eSVM performs better than RF during the 5 fold CV, so we next choose it to train a final model on the whole dataset without a training/testing set division. Then, this model can be used on external datasets to predict the unknown sample labels.

This is fulfilled via the function `maintrain`.

```
mods <- maintrain(y.. = labels,
                  betas.. = betas,
                  subset.CpGs = 10000,
                  seed = 1234,
                  method = 'eSVM',
                  calibrationmethod = c('LR', 'FLR', 'MR'))
```

We set the parameter `calibrationmethod` with the vector `c('LR', 'FLR', 'MR')`, so that all of the 3 calibration methods will be used and the 3 calibrated models, as well as the raw model will be returned with the result `mods`.

```
summary(mods)
#>                     Length Class      Mode
#> mod                    3   -none-     list
#> rawscores           2667   -none-     numeric
#> platt.calfits          7   -none-     list
#> probs.lr            2667   -none-     numeric
#> platt.brglm.calfits    7   -none-     list
#> probs.flr          2667   -none-    numeric
#> glmnet.calfit        12   cv.glmnet list
#> probs.mr           2667   -none-    numeric
```

You can see `mods` contains 8 slots and the one named 'mod' is the raw model trained from the whole dataset, while the ones named 'platt.calfits', 'platt.brglm.calfits', and 'glmnet.calfit' are the 3 calibrated models of 'LR', 'FLR', and 'MR'. Other slots are the prediction scores of the models on the whole dataset.

Then, another function `mainpredict` can be used, and the different slots of `mods`, as well as the external data you want to used to predict their sample labels can be transferred to this function to get the labels. More details can be found at the help document for `mainpredict`.

This package also contains other functions, such as multi-omics data classification and visualization (functions `maincv` and `mainJvisR`), DBSCAN clustering (function `clustering`), etc. We will not cover them in this tutorial to make it more simplified, and their details can be found in the help documents.

References

1. Capper, D., Jones, D.T.W., Sill, M., Hovestadt, V., Schrimpf, D., Sturm, D., Koelsche, C., Sahm, F., Chavez, L., Reuss, D.E., et al. (2018). DNA methylation-based classification of central nervous system tumours. Nature 555, 469-474.
2. Koelsche, C., Schrimpf, D., Stichel, D., Sill, M., Sahm, F., Reuss, D.E., Blattner, M., Worst, B., Heilig, C.E., Beck, K., et al. (2021). Sarcoma classification by DNA methylation profiling. Nature Communications 12, 498.
3. Liang, S., Mohanty, V., Dou, J., Miao, Q., Huang, Y., Müftüoğlu, M., Ding, L., Peng, W., and Chen, K. (2021). Single-cell manifold-preserving feature selection for detecting rare cell populations. Nature Computational Science 1, 374-384.
4. Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., and Smyth, G.K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Res 43, e47.