User's Guide

Asad Khan[1], Sajid Shah*[1], Fazli Wahid[2], Fiaz Gul Khan[1] and Saima Jabeen[1]

[1]Department of Computer Science COMSATS Institute of IT, Abbottabad 22060, Pakistan
[2] Department of Environmental Sciences COMSATS Institute of IT, Abbottabad 22060, Pakistan.

May 22, 2017

Contents

*sajidshah@ciit.net.pk

1 Introduction

We aimed to write this user's guide to help the research community to generate the results presented in our paper with title **'MicroRNA precursors identification using reduced and hybrid features'**. It could be considered as alternative to the online server which are established by many researchers to serve the research community.

In order to re-generate results using our proposed technique, the user must install R language. A part from fundamental R installation the user must also install the 'CMA' package of **bioconductor**. Other required packages of R are mentioned in their corresponding sections.

2 Dataset Construction

The dataset used in this research was taking from : `http://bioinformatics.hitsz.edu.cn/iMiRNA-PseDPC/data`

3 Feature Extraction

3.1 Distance pair composition

Distance pair composition features were extracted using repRNA Webserver. The link for the mentioned webserver is: `http://bioinformatics.hitsz.edu.cn/repRNA/`

3.2 Triplet Element

Triplet features were extracted using repRNA Webserver. The corresponding link is: `http://bioinformatics.hitsz.edu.cn/repRNA/`

3.3 di-nucleotide Frequency

Dinucleotide Frequency features were extracted using Biostrings package of Bioconductor. The following line of R code generates di-nucleotide frequency. The user must use the same parameters.

Signature of the function:

```
dinucleotideFrequency(x, width, step=1, as.prob=FALSE, as.array=FALSE,
fast.moving.side="right", with.labels=TRUE, simplify.as="matrix")
```

We have used the above function with following parameters.

```
dinucleotideFrequency(dataset, as.prob=TRUE)
```

3.4 tri- nucleotide Frequency

Trinucleotide Frequency features were extracted using Biostrings package of Bioconductor.

Signature of the function is:

```
trinucleotideFrequency(x, width, step=1,as.prob=FALSE, as.array=FALSE,
fast.moving.side="right", with.labels=TRUE, simplify.as="matrix")
```

We have used the above function with following parameters.

```
trinucleotideFrequency(dataset, as.prob=TRUE)
```

3.5 tetra-nucleotide Frequency

Tetranucleotide Frequency features were extracted using Biostrings package of Bioconductor.

Signature of the function:

```
oligonucleotideFrequency(x, width, step=1,as.prob=FALSE, as.array=FALSE,
fast.moving.side="right", with.labels=TRUE, simplify.as="matrix")
```

We have used the above function as follows:

```
oligonucleotideFrequency(dataset,4, as.prob=TRUE)
```

4 Dimension Reduction

4.1 Principle Component Analysis (PCA)

Dimension was reduced using PCA algorithm of **caret** package in R.

```
svmRes <- train(label~., whole, method = "svmRadial", preProcess=c("pca"),
trControl = trainControl(method = "cv",number=5))

%preprocessing as well as classification

fusionMatrix(svmRes,svmRes$label)

rfRes <- train(label~., whole, method = "rf", preProcess=c("pca"),
trControl= trainControl(method = "cv",number=5))

%preprocessing as well as classification

fusionMatrix(rfRes,rfRes$label)
```

4.2 Partial Least Square (PLS)

Dimension was reduced using pls algorithm of **plsgenomics** package in R.

```
output.pls <- pls.regression(cbind(dpc,triplet,x2,x3,x4),
transformy(label), ncomp=40)
```

where dpc is distance pair composition, x2, x3 and x4 are di, tri, tetra nucleotide frequencies. $ncomp = 40$ is set for SVM. In case of Radom Forest best results were achieved by setting $ncomp = 06$.

```
data.learn <- scale(cbind(dpc,triplet,x2,x3,x4),scale=FALSE,
center=output.pls$meanX)%*%output.pls$R
```

5 Classification

5.1 Support Vector Machine (SVM)

We have used radial svm kernel. The below code is written in CMA package in Bioconductor.

```
lset <- GenerateLearningsets(y=label, method = "CV", fold=5)

%five cross validation

svm <- classification(data.learn, label, learningsets = lset,
 classifier=svmCMA,models=FALSE)

ftable(join(svm))

%data.learn contained preprocessed data of
partial least square regression
```

The above code uses five cross validation technique. In order to use jackknife as validation technique use the following line of code.

```
lset <- GenerateLearningsets(y=label, method = "LOOCV")
```

5.2 Random Forest (RF)

We have used the Random forest classifier of CMA Package of bioconductor.

```
lset <- GenerateLearningsets(y=label, method = "CV", fold=5)
%five cross validation

rf <- classification(data.learn, label, learningsets = lset,
classifier=rfCMA,models=FALSE)
```

```
ftable(join(rf))

%data.learn contained preprocessed data of
partial least square regression
```

The above code uses five cross validation technique. In order to use jackknife as validation technique use the following line of code.

```
lset <- GenerateLearningsets(y=label, method = "LOOCV")
```

6 Validation

6.1 Five Cross Validation

6.2 Jackknife

7 R code

Copy and paste the following line function in a file and give it a name with *.R extension. Also copy the file with name .... containing the micro RNA features (see detail in the paper), already processed by PLS. Set the file path in the bellow function and run it in the R environment. The following line of code uses only SVM classifier, for Random Forest, use the instructions mentioned in this user's guide.

```
SVMmicroRNApredict<-function(plsProcessedFile=
"PreProcessedData_PLS.csv",
m="CV", f=5)
{
library("CMA")
data<-read.csv(plsProcessedFile)
data<-as.matrix(data)
label<-c(rep(1,1612),rep(-1,1612))
if(m=="CV")
lset <- GenerateLearningsets(y=label, method = m, fold=f)
else
lset <- GenerateLearningsets(y=label, method = "LOOCV")
Results <- classification(data, label, learningsets = lset,
classifier =svmCMA,models=FALSE)
ftable(join(Results))
}
```

It will generate the following output after running the following line of code.

```
> source('SVMmicroRNApredict.R')
> SVMmicroRNApredict()
```

OutPut:

```
iteration 1
iteration 2

WARNING: reaching max number of iterations
iteration 3

WARNING: reaching max number of iterations
iteration 4

WARNING: reaching max number of iterations
iteration 5
number of missclassifications:  215
missclassification rate:  0.067
sensitivity: 0.924
specificity: 0.942
predicted
true    0    1
0 1519   93
1  122 1490
```