Mega Learning

Feng Zeng
fengzeng@scarletmail.rutgers.edu
Department of Statistics and Biostatistics
Rutgers University

Jia Song
js1582@scarletmail.rutgers.edu
Department of Statistics and Biostatistics
Rutgers University

December 23, 2016

Abstract

SVM, neural network, glmnet etc. have been widely used in machine learning field. The packages of these algorithms also have been built and modified over time. When facing with a new data set of classification problem, people always try to apply many methods to find out which one is the most appropriate and best. This *megaLearn* package provides a convenient way to take data set as input and gave the result of the algorithms that you want to try. Totally 9 algorithms were included in the package:

- random forest
- SVM
- Ada boosting
- CART
- glmnet
- knn
- PCA-LDA
- neural network
- Naive Bayes

1 Usage in R

This package is basically a classification package. However, it can also be used build regression models if you choose the appropriate algorithms. *MegaLearn()* has a "*split_ ratio*" argument which is useful when you want

to use the training and testing set method. *MegaCV()* has a "*nfold*" argument for those who want to build a cross validation model. The algorithm specified in "*method*" argument will be returned with the model. Algorithms specified in "*methodslist*" argument will only be returned with accuracy for comparison purpose.

2 Examples

2.1 Weight Calculations

In this package, we give two methods to calculate p-values: one for continuous explanatory variable and one for dummy variables. Then package "*qvalues*" is used to calculate qvalues. logarithm of qvalues were used as the final weights.

2.1.1 Dummy explanatory variables

"geno" data is used to demonstrate here. All categorical variables has to be converted to dummy format first.

```
library(megaLearn)
randomNum <- sample.int(dim(xdata)[1])
ranGenoData <- xdata[randomNum,]
ynew <- y[randomNum]
xtrain <- ranGenoData[1:48, ]
xtest <- ranGenoData[49:72, ]
ytrain <- ynew[1:48, ]
ytest <- ynew[49:72, ]
pvs <- pvalue1(xtrain, ytrain)
```

pvs will be a vector of p-values, corresponding to the values from fisher tests.

2.1.2 Continuous explanatory variables

The sample data that included in the package is used in this example. In this data, "x" is the "bag of words" of Hilary Clinton and Donald Trump's tweets text content. "y" is a vector of character indicating which person the corresponding tweets belongs to.

```
library(megaLearn)
pvs2 <- pvalue2(x, y)
```

pvs2 is a vector of p values, corresponding to the values from ANOVA.

2.1.3 Weights

The "*weight*" function in the package take a numeric vector (pvalues) as input, gives a numeric vector as output, which is the weight we want. "pvs2", which was got from the previous part will be used as input here.

```
library(megaLearn)
wt <- weight(pvs2)
```

2.2 Training and Testing Set Split

Some users prefer using Training and Testing setting to evaluate the efficiency of classification. The function *split_data* was included in this package. Once the data has been imported to R, user can call the function to get xtrain, ytrain, xtest, and ytest data frame. Here we use "HCDT.data" as example, once the "HCDT.data"was imported into R, and we got the data frame of x and y, we can apply the splitting procedure.

```
df=split_data(x,y,split_ratio=0.25,
                  weight=NULL, seed=111)
xtrain=df$xtrain
ytrain=df$ytrain
xtest=df$xtest
ytest=df$ytest
```

2.3 Mega Learning Given Training and Testing Split Ratio

The function *MegaLearning* can simultaneously apply nine algorithms for the learning tasks, with input of data frame of x and y, the split ratio, and specified parameters for each algorithms).

2.3.1 Usage

```
MegaLearning=function(x,y, method = NULL,
    methodslist=NULL,
    rfntree = 500, rfmtry = NULL,
    svmkernel = "linear", svmscale = FALSE,
    adamfinal = 100,
    glmnetalpha = 0.9, glmnetfamily = "multinomial",
    glmnetstandardize = FALSE,
    knnk = 1,
    pcatol = 0.2, pcascale = FALSE, pcacenter = FALSE,
    nnetsize = 5, nnetrang = 0.1, nnetdecay = 0.1,
    nnetMaxNWts = 5000, nnetlinout = FALSE,
    nblap = 1,
    split_ratio=0.25,
    seed=164000082,
    weight=NULL)
```

2.3.2 Parameters

- x: data frame of x
- y: data frame of y
- method: one specified algorithm, for which need return the fitted model,and prediction accuracy
- methodslist: at most nine algorithms, for which, will return a table of methods accuracy

- `split_ratio`: the ratio for splitting the data into training and testing.
- seed: the number as seed for any random procedure during the learning
- weight: a vector of weights for each variable

- random forest
  - rfntree: number of trees to be built
  - rfmtry: Number of variables randomly sampled as candidates at each split
- svm
  - svmkernel: the kernel used in training and predicting, can be "linear","polynomial","radial based", or "sigmod"
  - svmscale: A logical vector indicating the variables to be scaled. If weight is used, set *svmscale*=FALSE.
- ada boosting
  - admfinal: the number of iterations or the number of trees to use.
- glmnet
  - glmnetalpha: The elasticnet mixing parameter, with 0 to 1.
  - glmnetfamily: Response type, can be "*binomial*","*multinomial*", and so on.
  - glmnetstandardize: Logical indicator for x variable standardization, if weight is used, set *glmnetstandardize*=FALSE.
- knn
  - knnk: number of neighbours considered during modeling and prediction
- pca-lda
  - pcatol: a value indicating how many components should be retained
  - pcascale: a logical value indicating whether the variables should be scaled. If weight is used, set *pcascale*=FALSE.
  - pcacenter: indicating whether the variables should be shifted to be zero centered
- nnet
  - nnetsize: number of units in the hidden layer
  - nnetrang: initial random weights
  - nnetdecay: parameter for weight decay
  - nnetMaxNWts: maximum number of iterations
  - nnetlinout: switch for linear output units
- naive bayes
  - nblap: positive number to control the laplace smoothing

2.3.3 Example

```
library(megaLearn)
test1=MegaLearning(x,y,method="NNET",
        methodslist=c("SVM","PCA-LDA"),
        split_ratio=0.25,seed=111)

test1$method_accuracy;
$accuracy
[1] 0.8293706

test1$methods_accuracy
       SVM    PCA-LDA
0.8706294 0.4755245
```

2.4 Mega Learning Given Cross Validation Fold

2.4.1 Usage

```
MegaCV=function(x,y, method = NULL,
    methodslist=NULL,
    nfold=3,
    rfntree = 500, rfmtry = NULL,
    svmkernel = "linear", svmscale = FALSE,
    adamfinal = 100,
    glmnetalpha = 0.9, glmnetfamily = "multinomial",
    glmnetstandardize = FALSE,
    knnk = 1,
    pcatol = 0.2, pcascale = FALSE, pcacenter = FALSE,
    nnetsize = 5, nnetrang = 0.1, nnetdecay = 0.1,
    nnetMaxNWts = 5000, nnetlinout = FALSE,
    nblap = 1,
    seed=164000082,
    weight=NULL)
```

2.4.2 Parameters

Most parameters for function *MegaCV* are the same as function *MegaLearning*, except *MegaCV* will have the parameter *nfold*, while *MegaLearning* has the parameter *split_data*.

nfold: the number of folds for the cross validation setting

2.4.3 Example

```
library(megaLearn)
test2=MegaCV(x,y,method="NNET",methodslist=c("SVM","PCA-LDA"),
nfold=3,seed=111)

test2$model
test2$method_accuracy
      NNET
```

```
0.8372948

test2$methods_accuracy
      SVM     PCA–LDA
0.8640328  0.4548838
```

References

[1] *e1071*: https://cran.r-project.org/web/packages/e1071/e1071.pdf

[2] *class*: https://cran.r-project.org/web/packages/class/class.pdf

[3] *adabag*: https://cran.r-project.org/web/packages/adabag/adabag.pdf

[4] *glmnet*: https://cran.r-project.org/web/packages/glmnet/glmnet.pdf

[5] *rpart*: https://cran.r-project.org/web/packages/rpart/rpart.pdf

[6] *MASS*: https://cran.r-project.org/web/packages/MASS/MASS.pdf