Package 'softmaxreg'

September 9, 2016

Type Package

Title Training Multi-Layer Neural Network for Softmax Regression and Classification

Version 1.2

Date 2016-09-08

Author Xichen Ding <rockingdingo@gmail.com>

Maintainer Xichen Ding <rockingdingo@gmail.com>

Depends R (>= 2.10)

Imports methods

Description Implementation of 'softmax' regression and classification models with multiple layer neural network. It can be used for many tasks like word embedding based document classification, 'MNIST' dataset handwritten digit recognition and so on. Multiple optimization algorithm including 'SGD', 'Adagrad', 'RMSprop', 'Moment', 'NAG', etc are also provided.

License GPL (>= 2)

Repository CRAN

Repository/R-Forge/Project softmaxreg

Repository/R-Forge/Revision 6

Repository/R-Forge/DateTimeStamp 2016-09-09 05:30:53

Date/Publication 2016-09-09 12:37:04

NeedsCompilation no

R topics documented:

AIC.softmax 2
BIC.softmax 2
convertClass2Matrix 3
document 4
loadURLData 5
softmax-class 5
softmaxReg 6
trainModel 12

word2vec 13
wordEmbed 15

Index **16**

`AIC.softmax` *Calculate AIC of Fitted Softmax Regression Model*

Description

Caculate akaike information criterion of fitted softmax regression model

Usage

```
## S3 method for class 'softmax'
AIC(object, ...)
```

Arguments

`object`	A object of "softmax" returned by `softmaxReg` function
`...`	Other arguments

Value

Numeric Value of AIC

See Also

`BIC.softmax` `softmaxReg`

`BIC.softmax` *Calculate BIC of Fitted Softmax Regression Model*

Description

Caculate bayesian information criterion of fitted softmax regression model

Usage

```
## S3 method for class 'softmax'
BIC(object, ...)
```

Arguments

`object`	A object of "softmax" returned by `softmaxReg` function
`...`	Other arguments

Value

Numeric Value of BIC

See Also

AIC.softmax softmaxReg

---

convertClass2Matrix *Convert A Vector of Factors to Matrix*

Description

Convert A Vector of Factors to Matrix

Usage

```
convertClass2Matrix(target)
```

Arguments

target	Vector of factor representing each class.

Details

This Function can be used to convert factor to matrix yMat, e.g. For type 'raw' softmaxReg function input yMat, softmax regression.

Value

Matrix with dimensions number of observation * number of class factors

See Also

softmaxReg

Examples

```
## This Function can be used to convert factor to matrix yMat.
## e.g. For type 'raw' softmaxReg function input yMat, softmax regression.
y = as.factor(c(rep(1,50),rep(2,50),rep(3,50)))
yMat = convertClass2Matrix(y)
nObs = dim(yMat)[1]
K = dim(yMat)[2]
nObs
K
```

document *Easy Implementation to Read Multiple Documents within the Folder*

Description

Easy implementation to read multiple documents within the folder with extension pattern.

Usage

```
document(path, name = NULL, pattern = "txt")
```

Arguments

path	Character Vectors representing the folders' path. One element of string denotes reading the document from one folder and a vector of characters denotes reading the documents from multiple folders simultaneously.
name	Character representing the name of the specific file to read. Default NULL. If NULL, function will read all the text files in that folder.
pattern	Character for the file extensions of the text files, like "txt", "csv", etc. Default "txt".

Value

Vectors of characters, each element in the vector contains the text of one file.

See Also

loadURLData wordEmbed

Examples

```
## Not run:
path = "your_local_path"
docs = document(path, pattern = 'txt')

## End(Not run)
```

loadURLData *Download and Unzip Web Datasets*

Description

Download web datasets from URL to local path and unzip the data.

Usage

```
loadURLData(URL, folder, unzip = FALSE)
```

Arguments

URL	String of the url of the web dataset.
folder	String of the path of the folder you want to put the downloaded files. Folder path will be set as the working directory.
unzip	Boolean variable. If true, the ".zip" files will be unzipped.

See Also

document wordEmbed

Examples

```
## Not run:
# Download UCI Archived Dataset from URL:
# http://archive.ics.uci.edu/ml/machine-learning-databases/00217/C50.zip
# Reuter 50 DataSet
URL = "http://archive.ics.uci.edu/ml/machine-learning-databases/00217/C50.zip"
folder = getwd()
loadURLData(URL, folder, unzip = TRUE)

## End(Not run)
```

softmax-class *Class* "softmax"

Description

"softmax" class returned by softmaxReg function for softmax regression

Objects from the Class

Objects can be created by calls of the form new("softmax", ...).

Slots

`weights`: Object of class "`list`"

`data`: Object of class "`list`"

`K`: Object of class "`numeric`"

`iteration`: Object of class "`numeric`"

`loss`: Object of class "`numeric`"

`fitted.values`: Object of class "`matrix`"

`convergence`: Object of class "`logical`"

`type`: Object of class "`character`"

`funName`: Object of class "`character`"

Methods

$ `signature(x = "softmax")`: ...

Examples

```
showClass("softmax")
```

`softmaxReg` *Fit Multi-Layer Softmax Regression or Classification Model*

Description

Fit softmax regression or classification model with multiple hidden layers neural networks and final softmax layer.

Usage

```
softmaxReg(x, ...)

## Default S3 method:
softmaxReg(x, y, hidden = c(), funName = 'sigmoid', maxit = 3000,
rang = 0.1, type = "class", algorithm = "rmsprop", rate = 0.05,
L2 = FALSE, penalty = 1e-4, threshold = 1e-4, batch = 50,...)

## S3 method for class 'formula'
softmaxReg(formula, data, hidden = c(), funName = 'sigmoid', maxit = 3000,
rang = 0.1,type = "class", algorithm = "rmsprop", rate = 0.05,
L2 = FALSE, penalty = 1e-4, threshold = 1e-4, batch = 50,...)

## S3 method for class 'softmax'
predict(object, newdata, ...)

## S3 method for class 'softmax'
summary(object, ...)
```

Arguments

formula	Formula of form y ~ x1 + x2 + ... for 'class' type classification; And (y1 + y2+ ...+ yk) ~ x1 + x2 + ... for 'raw' type regression.
x	Matrix or data frame of x input values.
y	Vector of target values y for 'class' type classfication and matrix or data frame of target values (y1,y2,...yk) for 'raw' type regression.
data	Data frame containing the variables in formula.
hidden	Numeric vector of integers specifying the number of hidden nodes in each layer, e.g. hidden = c(8,5,...). Default NULL.
funName	Name of neural network activation function, including 'sigmoid', 'tanh', 'relu'. Default 'sigmoid'.
maxit	Integer for maximum number of iterations. Default 3000.
rang	Parameter for the range of initial random weights [-rang, rang]. Default 0.1.
type	Parameter indicating the type of softmax task: 'class' denotes the softmax classfication model and the fitted values are factors; 'raw' denotes softmax regression model and the fitted values are raw probability or percentage data of each group. Default 'class'.
algorithm	Parameter indicating which gradient descenting learning algorithm to use, including 'sgd', 'adagrad', 'rmsprop', 'adadelta', 'momentum', 'nag'(Nesterov Momentum), etc. Default 'rmsprop'.
rate	Parameter for the initial learning rate. Default 0.05.
L2	Boolean variable indicating whether L2 regularization term is added to the loss function and gradient to prevent overfitting. Default FALSE.
penalty	Parameter for the penalty cost of the L2 regularization term if L2 is TRUE. Default 1e-4.
threshold	Parameter for the threshold of iteration convergence: loss value less than threshold. Default 1e-4.
batch	Parameter for mini-batch size. Default 50.
object	An object of class "softmax", the fitted model of softmaxReg function.
newdata	Matrix or dataframe of new Data for prediction.
...	Other arguments

Details

This function can be used to train typical n-class classification models. Also, it can be used to fit 'raw' data regression, e.g. the percentage/probability data of each group in the Multinomial Logit/Probit model, as well.

Value

object of class "softmax"

weights	Optimal weights parameters found by softmax model, including list of W and B for all layers.

data	Input Training Data.
K	Number of K groups fitted by softmax model.
loss	Numeric vector of the loss function values over iterations.
fitted.values	Matrix of the fitted values yFitMat for the training data. Dimensions: number of observations by K;
iteration	Number of iteration reached before stop.
convergence	Boolean variable for whether softmax model reached convergence.

Author(s)

Xichen Ding

References

MNIST Dataset HandWritten Digit Recognition: http://yann.lecun.com/exdb/mnist/

MNIST Data Reading method reuse R code from: brendan o'connor - https://gist.github.com/brendano/39760

Reuter 50 DataSet: UCI Archived Dataset: http://archive.ics.uci.edu/ml/machine-learning-databases/00217/C50.zip

See Also

wordEmbed document loadURLData

Examples

```
## Not run:
#### Example 1, Softmax classification with hidden layer and no regularization term

library(softmaxreg)
data(iris)
x = iris[,1:4]
y = iris$Species
# Training with hidden layer set 5 units
softmax_model = softmaxReg(x, y, hidden = c(5), maxit = 100, type = "class",
  algorithm = "adagrad", rate = 0.05, batch = 20)
summary(softmax_model)
yFitMat = softmax_model$fitted.values
yFit = c()
for (i in 1:length(y)) {
yFit = c(yFit, which(yFitMat[i,]==max(yFitMat[i,])))
}
table(y, yFit)
# Caculate AIC and BIC information criterion
aic = AIC(softmax_model)
bic = BIC(softmax_model)
cat("AIC",aic,'\n')
cat("BIC",bic,'\n')

# Make new Prediction
newdata = iris[1:100,1:4]
```

```
yPred = predict(softmax_model, newdata)

#### Example 2, Softmax classification with formula and dataframe input

f = formula(Species~.) # formula with succinct expression
softmax_model_fm = softmaxReg(f, data = iris, hidden = c(5), maxit = 100, type = "class",
  algorithm = "adagrad", rate = 0.05, batch = 20)
summary(softmax_model_fm)

#### Example 3: Softmax classfication with L2 regularization

softmax_model_L2 = softmaxReg(x, y, hidden = c(5), maxit = 100, type = "class",
  algorithm = "adagrad", L2 = TRUE, penalty = 1e-4, batch = 20)
summary(softmax_model_L2)

# Compare Two Model Loss
# Note L2 loss value include the ||W||^2 term, larger than loss of previous model
loss1 = softmax_model$loss
loss2 = softmax_model_L2$loss
plot(c(1:length(loss1)), loss1, xlab = "Iteration", ylab = "Loss Function Value",
  type = "l", col = "black")
lines(c(1:length(loss2)), loss2, col = "red")
legend("topright", c("Loss 1: No Regularization", "Loss 2: L2 Regularization"),
  col = c("black", "red"),pch = 1)


#### Example 4: Compare different learning algorithms 'adagrad','sgd',
# 'rmsprop', 'momentum', 'nag' (Nesterov Momentum)

library(softmaxreg)
data(iris)
x = iris[,1:4]
y = iris$Species
model1 = softmaxReg(x, y, hidden = c(), funName = 'sigmoid', maxit = 100, rang = 0.1,
  type = "class", algorithm = "sgd", rate = 0.1, batch = 150)
loss1 = model1$loss

model2 = softmaxReg(x, y, hidden = c(), funName = 'sigmoid', maxit = 100, rang = 0.1,
  type = "class", algorithm = "adagrad", rate = 0.1, batch = 150)
loss2 = model2$loss

model3 = softmaxReg(x, y, hidden = c(), funName = 'sigmoid', maxit = 100, rang = 0.1,
  type = "class", algorithm = "rmsprop", rate = 0.1, batch = 150)
loss3 = model3$loss

model4 = softmaxReg(x, y, hidden = c(), funName = 'sigmoid', maxit = 100, rang = 0.1,
  type = "class", algorithm = "momentum", rate = 0.1, batch = 150)
loss4 = model4$loss

model5 = softmaxReg(x, y, hidden = c(), funName = 'sigmoid', maxit = 100, rang = 0.1,
  type = "class", algorithm = "nag", rate = 0.1, batch = 150)
loss5 = model5$loss
```

```
# plot the loss convergence
iteration = c(1:length(loss1))
plot(iteration, loss1, xlab = "iteration", ylab = "loss", ylim = c(0,
   max(loss1,loss2,loss3,loss4,loss5) + 0.01), type = "p", col = "black", cex = 0.7)
title("Convergence Comparision Between Learning Algorithms")
points(iteration, loss2, col = "red", pch = 2, cex = 0.7)
points(iteration, loss3, col = "blue", pch = 3, cex = 0.7)
points(iteration, loss4, col = "green", pch = 4, cex = 0.7)
points(iteration, loss5, col = "magenta", pch = 5, cex = 0.7)

legend("topright", c("SGD", "Adagrad", "RMSprop", "Momentum", "NAG"),
    col = c("black", "red", "blue", "green", "magenta"),pch = c(1,2,3,4,5))

## Comments: From this experiments we can see that momemtum learning algorithm
## generally converge faster than the standard sgd and its variations


#### Example 5: Multiple class classification: Read Online Dataset and make document classification

library(softmaxreg)
data(word2vec) # default 20 dimension word2vec dataset
#### Reuter 50 DataSet UCI Archived Dataset from
## URL: "http://archive.ics.uci.edu/ml/machine-learning-databases/00217/C50.zip"
URL = "http://archive.ics.uci.edu/ml/machine-learning-databases/00217/C50.zip"
folder = getwd()
loadURLData(URL, folder, unzip = TRUE)

##Training Data
subFoler = c('AaronPressman', 'AlanCrosby', 'AlexanderSmith', 'BenjaminKangLim', 'BernardHickey')
docTrain = document(path = paste(folder, "\C50train\",subFoler, sep = ""), pattern = 'txt')
xTrain = wordEmbed(docTrain, dictionary = word2vec)
yTrain = c(rep(1,50), rep(2,50), rep(3,50), rep(4,50), rep(5,50))
# Assign labels to 5 different authors

##Testing Data
docTest = document(path = paste(folder, "\C50test\",subFoler, sep = ""), pattern = 'txt')
xTest = wordEmbed(docTest, dictionary = word2vec)
yTest = c(rep(1,50), rep(2,50), rep(3,50), rep(4,50), rep(5,50))
samp = sample(250, 50)
xTest = xTest[samp,]
yTest = yTest[samp]

## Train Softmax Classification Model, 20-10-5
softmax_model = softmaxReg(xTrain, yTrain, hidden = c(10), maxit = 500, type = "class",
    algorithm = "nag", rate = 0.1, L2 = TRUE)
summary(softmax_model)
yFit = predict(softmax_model, newdata = xTrain)
table(yTrain, yFit)

## Testing
yPred = predict(softmax_model, newdata = xTest)
table(yTest, yPred)
```

```
#### Comments: Increase the word2vec dimensions to 50 or even 100 will help increase
#### the capacity of the model and prediction precision

#### Example 6: 'MNIST' dataset HandWritten Digit Recognition
## Download MNIST Dataset from below URL and Gunzip them
## http://yann.lecun.com/exdb/mnist/
## MNIST Data Reading method reuse R code from:
## brendan o'connor - https://gist.github.com/brendano/39760

library(softmaxreg)
# Replace with your local path
path = "D:\DeepLearning\MNIST\"

## 10-class classification, Digit 0-9
x = load_image_file(paste(path,'train-images-idx3-ubyte', sep=""))
y = load_label_file(paste(path,'train-labels-idx1-ubyte', sep=""))
xTest = load_image_file(paste(path,'t10k-images-idx3-ubyte',sep=""))
yTest = load_label_file(paste(path,'t10k-labels-idx1-ubyte', sep=""))

## Normalize Input Data
x = x/255
xTest = xTest/255

## Compare Convergence Rate of MNIST dataset
model1 = softmaxReg(x, y, hidden = c(), funName = 'sigmoid', maxit = 50, rang = 0.1,
    type = "class", algorithm = "sgd", rate = 0.01, batch = 100)
loss1 = model1$loss
model2 = softmaxReg(x, y, hidden = c(), funName = 'sigmoid', maxit = 50, rang = 0.1,
    type = "class", algorithm = "adagrad", rate = 0.01, batch = 100)
loss2 = model2$loss
model3 = softmaxReg(x, y, hidden = c(), funName = 'sigmoid', maxit = 50, rang = 0.1,
    type = "class", algorithm = "rmsprop", rate = 0.01, batch = 100)
loss3 = model3$loss
model4 = softmaxReg(x, y, hidden = c(), funName = 'sigmoid', maxit = 50, rang = 0.1,
    type = "class", algorithm = "momentum", rate = 0.01, batch = 100)
loss4 = model4$loss
model5 = softmaxReg(x, y, hidden = c(), funName = 'sigmoid', maxit = 50, rang = 0.1,
    type = "class", algorithm = "nag", rate = 0.01, batch = 100)
loss5 = model5$loss

# plot the loss convergence
iteration = c(1:length(loss1))
myplot = plot(iteration, loss1, xlab = "iteration", ylab = "loss", ylim = c(0,
    max(loss1,loss2,loss3,loss4,loss5) + 0.01), type = "p", col = "black", cex = 0.7)
title("Convergence Comparision Between Learning Algorithms")
points(iteration, loss2, col = "red", pch = 2, cex = 0.7)
points(iteration, loss3, col = "blue", pch = 3, cex = 0.7)
points(iteration, loss4, col = "green", pch = 4, cex = 0.7)
points(iteration, loss5, col = "magenta", pch = 5, cex = 0.7)

legend("topright", c("SGD", "Adagrad", "RMSprop", "Momentum", "NAG"),
    col = c("black", "red", "blue", "green", "magenta"),pch = c(1,2,3,4,5))
```

```
save.image()


## End(Not run)
```

`trainModel` *Train softmax regression and classification model*

Description

This function implements a feedforward neural networks with multiple hidden layers and a softmax final layer. For classification, set type as "class" and y as factor vector; for regression, set type as "raw" and y as matrix with dimension nObs * K. K denotes the group number.

Usage

```
trainModel(x, y, hidden, funName, maxit, rang, type, algorithm, rate,
    L2, penalty, threshold, batch)
```

Arguments

`x`	matrix or data frame of x input values.
`y`	vector of target values for 'class' type classfication and matrix or data frame of target values for 'raw' type regression.
`hidden`	vector of integers specifying the number of hidden nodes in each layer.
`funName`	activation function name of neuron, e.g. 'sigmoid', 'tanh', 'relu' etc. In default, it is set to 'sigmoid'.
`maxit`	maximum number of iterations.Default 3000.
`rang`	parameter for the range of initial random weights. Default 0.1 [-rang, rang].
`type`	parameter indicating the type of softmax task: "class" denotes the softmax classfication model and the fitted values are factors; "raw" denotes softmax regression model and the fitted values are the probability or percentage of each group. Default "class".
`algorithm`	parameter indicating which gradient descenting learning algorithm to use, including "sgd", "adagrad", "rmsprop", "adadelta", etc. Default "adagrad".
`rate`	parameter of learning rate. Default 0.05.
`L2`	Boolean variable indicating whether L2 regularization term is added to the loss function and gradient to prevent overfitting. Default FALSE.
`penalty`	Parameter for the penalty cost of the L2 regularization term if L2 is TRUE. Default 1e-4.
`threshold`	Parameter for the threshold of iteration convergence: loss value less than threshold. Default 1e-4.
`batch`	Parameter for mini-batch size. Default 50.

Value

object of class "softmax"

weights	Optimal weights parameters found by softmax model, including list of W and B for all layers.
data	Input Training Data.
K	Number of K groups fitted by softmax model.
loss	Numeric vector of the loss function values over iterations.
fitted.values	Matrix of the fitted values yFitMat for the training data. Dimensions: number of observations by K;
iteration	Number of iteration reached before stop.
convergence	Boolean variable for whether softmax model reached convergence.

See Also

softmaxReg

Examples

```
## Not run:
library(softmaxreg)
data(iris)
x = iris[,1:4]
y = iris$Species
sofmax_model = trainModel(x, y, hidden = c(5), funName = 'sigmoid', maxit = 3000,
    rang = 0.1, type = "class", algorithm = "adagrad", rate = 0.05, threshold = 1e-3)
summary(sofmax_model)
yFitMat = sofmax_model$fitted.values
yFit = c()
for (i in 1:length(y)) {
yFit = c(yFit, which(yFitMat[i,]==max(yFitMat[i,])))
}
table(y, yFit)

## End(Not run)
```

---

word2vec *Pre-trained Word2vec Dataset from Corpus*

Description

This dataset is a small pre-trained word2vec dataset with 20 dimensions and 5296 words.

Usage

```
data("word2vec")
```

Format

A data frame with 12853 observations on the following 21 variables.

`word` character

`col1` numeric

`col2` numeric

`col3` numeric

`col4` numeric

`col5` numeric

`col6` numeric

`col7` numeric

`col8` numeric

`col9` numeric

`col10` numeric

`col11` numeric

`col12` numeric

`col13` numeric

`col14` numeric

`col15` numeric

`col16` numeric

`col17` numeric

`col18` numeric

`col19` numeric

`col20` numeric

Source

Reuter_50_50 dataset,UCI Machine Learning Repository[https://archive.ics.uci.edu/ml/datasets/Reuter_50_50]

References

word2vec model are trained based on below text corpus:

Reuter_50_50 dataset,UCI Machine Learning Repository[https://archive.ics.uci.edu/ml/datasets/Reuter_50_50], Author: Zhi Liu, National Engineering Research Center For E-Learning Technology, Hubei Wuhan, China

Examples

```
data(word2vec)
```

wordEmbed *Embed Words to Vectors Using Pre-trained Word2vec Dictionary*

Description

Embed words in string to vectors using the pre-trained word2vec dictionary. User can also replace the word2vec dataframe with customized data.

Usage

```
wordEmbed(object, dictionary, meanVec)
```

Arguments

object	Vectors of text representing documents.
dictionary	Dataframe of pre-trained word2vec dataset. The First column is the word and the following columns are numeric vectors from word2vec models. The default dataset with the package is a pre-trained 20 dimension word2vec dataset.
meanVec	Boolean variable. If meanVec is TRUE, a matrix is returned with each row representing the mean of numeric vectors of all the words in a document. If FALSE, a list of matrix is returned in which each document is represented by a matrix.

Value

wordEmbed returns a matrix if meanVec is TRUE and a list of matrix if meanVec is FALSE.

See Also

document word2vec

Examples

```
data(word2vec) # load default 20 dimensions word2vec dataset
doc = "This is an example line of document"
docVectors = wordEmbed(doc, word2vec, meanVec = TRUE)
```

Index

∗Topic **\textasciitildekwd1**
    AIC.softmax, 2
    BIC.softmax, 2
    convertClass2Matrix, 3
    document, 4
    loadURLData, 5
    trainModel, 12
∗Topic **\textasciitildekwd2**
    AIC.softmax, 2
    BIC.softmax, 2
    convertClass2Matrix, 3
    document, 4
    loadURLData, 5
    trainModel, 12
∗Topic **classes**
    softmax-class, 5
∗Topic **datasets**
    word2vec, 13
$,softmax-method *(*softmax-class*)*, 5

AIC *(*AIC.softmax*)*, 2
AIC.softmax, 2, *3*

BIC *(*BIC.softmax*)*, 2
BIC.softmax, *2*, 2

convertClass2Matrix, 3

document, 4, *5*, *8*, *15*

loadURLData, *4*, 5, *8*

predict.softmax *(*softmaxReg*)*, 6

softmax-class, 5
softmaxReg, *2, 3*, 6, *13*
summary.softmax *(*softmaxReg*)*, 6

trainModel, 12

word2vec, 13, *15*
wordEmbed, *4, 5*, *8*, 15