Package 'LilRhino'

January 20, 2025

Type Package

Title For Implementation of Feed Reduction, Learning Examples, NLP and Code Management

Version 1.2.2

Author Travis Barton (2018)

Maintainer Travis Barton <travisdatabarton@gmail.com>

Description This is for code management functions, NLP tools, a Monty Hall simulator, and for implementing my own variable reduction technique called Feed Reduction. The Feed Reduction technique is not yet published, but is merely a tool for implementing a series of binary neural networks meant for reducing data into N dimensions, where N is the number of possible values of the response variable.

License GPL-2

Encoding UTF-8

Suggests textclean

Imports FNN, stringi, beepr, ggplot2, keras, dplyr, readr, parallel, tm, e1071, SnowballC, data.table, fastmatch, neuralnet

NeedsCompilation no

Repository CRAN

Date/Publication 2022-04-27 22:10:14 UTC

Contents

`Binary_Network` *Binary Decision Neural Network Wrapper*

Description

Used as a function of Feed_Reduction, Binary_Networt uses a 3 layer neural network with an adam optimizer, leaky RELU for the first two activation functions, followed by a softmax on the last layer. The loss function is binary_crossentropy. This is a keras wrapper, and uses tensorflow in the backend.

Usage

```
Binary_Network(X, Y, X_test, val_split, nodes, epochs, batch_size, verbose = 0)
```

Arguments

`X`	Training data.
`Y`	Training Labels. These must be binary.
`X_test`	The test Data
`val_split`	The validation split for keras.
`nodes`	The number of nodes in the hidden layers.
`epochs`	The number of epochs for the network
`batch_size`	The batch size for the network
`verbose`	Weither or not you want details about the run as its happening. 0 = silent, 1 = progress bar, 2 = one line per epoch.

Details

This function is a subset for the larger function Feed_Network. The output is the list containing the training and testing data converted into an approximation of probability space for that binary decision.

Value

`Train`	The training data in approximate probability space
`Test`	The testing data in 'double' approximate probability space

Author(s)

Travis Barton

References

Check out http://wbbpredictions.com/wp-content/uploads/2018/12/Redditbot_Paper.pdf and Keras for details

See Also

Feed_Network

Examples

```
## Not run:
if(8 * .Machine$sizeof.pointer == 64){
  #Feed Network Testing
  library(keras)
  library(dplyr)
    install_keras()
    dat <- keras::dataset_mnist()
    X_train = array_reshape(dat$train$x/255, c(nrow(dat$train$x/255), 784))
    y_train = to_categorical(dat$train$y, 10)
    X_test = array_reshape(dat$test$x/255, c(nrow(dat$test$x/255), 784))
    y_test =to_categorical(dat$test$y, 10)


    index_train = which(dat$train$y == 6 | dat$train$y == 5)
    index_train = sample(index_train, length(index_train))
    index_test = which(dat$test$y == 6 | dat$test$y == 5)
    index_test = sample(index_test, length(index_test))

    temp = Binary_Network(X_train[index_train,],
    y_train[index_train,c(7, 6)], X_test[index_test,], .3, 350, 30, 50)
  }

## End(Not run)
```

`Bootstrap_Data_Frame` *A function for bootstraping textual data so that all levels have the same number of entries.*

Description

This function takes a corpus and a set of labels and uses Bootstrap_Vocab to increase the size of each label until they are all the same length. Stop words are not bootstrapped.

Usage

```
Bootstrap_Data_Frame(text, tags, stopwords, min_length = 7, max_length = 15)
```

Arguments

`text`	text is the collection of textual data to bootstrap up.
`tags`	tags are the collection of tags that will be used to bootstrap. There should be one for every entry in 'text'. They do not have to be unique.
`stopwords`	stopwords to make sure are not apart of the bootstrapping process. It is advised to eliminate the most common words. See Stop_Word_Maker()
`min_length`	The shortest length allowable for bootstrapped words
`max_length`	The longest length allowable for bootstrapped words

Details

Most of the bootstrapped words will be nonseneical. The intention of this package is not to create new sentences, but to instead trick your model into thinking it has equal lengthed levels. This method is meant for bag of words style models.

Value

A data frame of your original documents along with the bootstrapped ones (column 1) along with their tags (column 2).

Author(s)

Travis Barton

Examples

```
test_set = c('I like cats', 'I like dogs', 'we love animals', 'I am a vet',
             'US politics bore me', 'I dont like to vote',
             'The rainbow looked nice today dont you think tommy')
test_tags = c('animals', 'animals', 'animals', 'animals',
             'politics', 'politics',
             'misc')

Bootstrap_Data_Frame(test_set, test_tags, c("I", "we"), min_length = 3, max_length = 8)
```

Bootstrap_Vocab *An internal function for Bootstrap_Data_Frame.*

Description

This function takes a selection of documents and bootstraps words from said sentences until there are N total sentences (both sudo and original).

Usage

```
Bootstrap_Vocab(vocab, N, stopwds, min_length = 7, max_length = 15)
```

Arguments

vocab	The collection of documents to boostrap.
N	The total amount of sentences to end up with
stopwds	A list of stopwords to not include in the bootstrapping proccess
min_length	The shortest allowable bootstrapped doument
max_length	The longest allowable bootstrapped document

Details

The min and max length arguements to not gaurantee that a sentence will reach that length. These senteces will be nonsensical.

Value

A vector of bootstrapped sentences.

Author(s)

Travis Barton

Examples

```
testing_set = c(paste('this is test',  as.character(seq(1, 10, 1))))

Bootstrap_Vocab(testing_set, 20, c('this'))
```

Codes_done *For announcing when code is done.*

Description

for alerting you when your code is done.

Usage

```
Codes_done(title, msg, sound = FALSE, effect = 1)
```

Arguments

title	The title of the notification
msg	The message to be sent
sound	Optional sound to blurt as well
effect	If sound it blurted, what should it be? (check beepr package for sound options)

Details

Only for Linix (as far as I know)

Author(s)

smacdonald (stack overflow) with modificaion by Travis Barton

References

https://stackoverflow.com/questions/3365657/is-there-a-way-to-make-r-beep-play-a-sound-at-the-end-of-a-script

Examples

```
Codes_done("done", "check it", sound = TRUE, effect = 1)
```

`Cross_val_maker` *For Creating a test and train set from a whole set*

Description

for making one dataset into two (test and train)

Usage

```
Cross_val_maker(data, alpha)
```

Arguments

`data`	matrix of data you want to split
`alpha`	the percent of data to split

Value

returns a list with accessable with the '$' sign. Test and Train are labeled as such.

Author(s)

Travis Barton

Examples

```
dat <- Cross_val_maker(iris, .1)
train <- dat$Train
test <- dat$Test
```

`Feed_Reduction` *A Function for converting data into approximations of probability space.*

Description

It takes the number of unique labels in the training data and tries to predict a one vs all binary neural network for each unique label. The output is an approximation of the probability that each individual input does not not match the label. Travis Barton (2018) http://wbbpredictions.com/wp-content/uploads/2018/12/Redditbot_Paper.pdf

Usage

```
Feed_Reduction(X, Y, X_test, val_split = .1,
               nodes = NULL, epochs = 15,
               batch_size = 30, verbose = 0)
```

Arguments

`X` Training data

`Y` Training labels

`X_test` Testing data

`val_split` The validation split for the keras, binary, neural networks

`nodes` The number nodes for the hidden layers, default is 1/4 of the length of the training data.

`epochs` The number of epochs for the fitting of the networks

`batch_size` The batch size for the networks

`verbose` Weither or not you want details about the run as its happening. 0 = silent, 1 = progress bar, 2 = one line per epoch.

Details

This is a new technique for dimensionality reduction of my own creation. Data is converted to the same number of dimensions as there are unique labels. Each dimension is an approximation of the probability that the data point is inside the a unique label. The return value is a list the training and test data with their dimensionality reduced.

Value

`Train` The training data in the new probability space

`Test` The testing data in the new probability space

Author(s)

Travis Barton.

References

Check out http://wbbpredictions.com/wp-content/uploads/2018/12/Redditbot_Paper.pdf for details on the proccess

See Also

Binary_Network

Examples

```
## Not run:
if(8 * .Machine$sizeof.pointer == 64){
#Feed Network Testing
library(keras)

  install_keras()
  dat <- keras::dataset_mnist()
  X_train = array_reshape(dat$train$x/255, c(nrow(dat$train$x/255), 784))
  y_train = dat$train$y
  X_test = array_reshape(dat$test$x/255, c(nrow(dat$test$x/255), 784))
  y_test = dat$test$y

  Reduced_Data2 = Feed_Reduction(X_train, y_train, X_test,
                              val_split = .3, nodes = 350,
                              30, 50, verbose = 1)

  library(e1071)
  names(Reduced_Data2$test) = names(Reduced_Data2$train)
 newdat = as.data.frame(cbind(rbind(Reduced_Data2$train, Reduced_Data2$test), c(y_train, y_test)))
  colnames(newdat) = c(paste("V", c(1:11), sep = ""))
  mod = svm(V11~., data = newdat, subset = c(1:60000),
           kernel = 'linear', cost = 1, type = 'C-classification')
  preds = predict(mod, newdat[60001:70000,-11])
  sum(preds == y_test)/10000

}

## End(Not run)
```

Load_Glove_Embeddings *Function for loading in pre-trained or personal word embedding softwares.*

Description

Loads in GloVes' pretrained 42 billion token embeddings, trained on the common crawl.

Usage

```
Load_Glove_Embeddings(path = 'glove.42B.300d.txt', d = 300)
```

Arguments

path	The path to the embeddings file.
d	The dimension of the embeddings file.

Details

The embeddings file should be the word, followed by numeric values, ending with a carriage return.

Value

The embeddings matrix.

Author(s)

Travis Barton

Examples

```
#This code only works if you have the 5g file found here: <https://nlp.stanford.edu/projects/glove/>

## Not run: emb = Load_Glove_Embeddings()
```

Monty_Hall *Monty Hall Simulator*

Description

A simulator for the famous Monty Hall Problem

Usage

```
Monty_Hall(Games = 10, Choice = "Stay")
```

Arguments

Games	The number of games to run on the simulation
Choice	Wether you would like the simulation to either 'Stay' with the first chosen door, 'Switch' to the other door, or 'Random' where you randomly decide to either stay or switch.

Details

This is just a toy example of the famous Monty Hall problem. It returns a ggplot bar chart showing the counts for wins or loses in the simulation.

Value

A ggplot graph is produced. There is no return value.

Author(s)

Travis Barton

Examples

```
Monty_Hall(100, 'Stay')
```

Nearest_Centroid *For performing the nearest centroid problem (with modifications) on MNST data specifically (general to come)*

Description

For Chen's homework, I'll change this when I generalize it.

Usage

```
Nearest_Centroid(X_train, X_test, Y_train)
```

Arguments

`X_train`	Training data
`X_test`	data to be tested
`Y_train`	training labels

Note

Based on homework from Guangling Chen's M251 class at SJSU

Author(s)

Travis Barton

Num_Al_Sep *Number/alpha numeric seperator for strings.*

Description

A Function for the separating of numbers from letters. 'b4' for example would be converted to 'b 4'.

Usage

```
Num_Al_Sep(vec)
```

Arguments

`vec`	The string vector in which you wish to separate the numbers from the letters.

Value

`output`	The separated vector.

Note

This is a really simple function really used inside other functions.

Author(s)

Travis Barton

Examples

```
test_vec = 'The most iconic American weapon has to be the AR15'
res = Num_Al_Sep(test_vec)
print(res)
```

`Percent` *Percent of confusion matrix*

Description

For finding the accuracy of confusion matricies with true/pred values

Usage

```
Percent(true, test)
```

Arguments

`true`	The true values
`test`	the test values

Details

Make sure your strings have the right values and create a square matrix.

Value

the percent acc.

Author(s)

Travis Barton

Examples

```
true <- rep(1:10, 10)
test <- rep(1:10, 10)
test[c(2, 22, 33, 89)] = 1
Percent(true, test)
#or
#percent(table(true, test))
```

Pretreatment *Pretreatment of textual documents for NLP.*

Description

This function goes through a number of pretreatment steps in preparation for vectorization. These steps are designed to help the data become more standard so that there are fewer outliers when training during NLP. The following effects are applied: 1. Non-alpha/numerics are removed. 2. Numbers are separated from letters. 3. Numbers are replaced with their word equivalents. 4. Words are stemmed (optional). 5. Words are lowercased (optinal).

Usage

```
Pretreatment(title_vec, stem = TRUE, lower = TRUE, parallel = FALSE)
```

Arguments

`title_vec`	Vector of documents to be pre-treated.
`stem`	Boolian variable to decide whether to stem or not.
`lower`	Boolian variable to decide whether to lowercase words or not.
`parallel`	Boolian variable to decide whether to run this function in parallel or not.

Details

This function returns a list. It should be able to accept any format that the function lapply would accept. The parallelization is done with the function Mcapply from the package 'parallel' and will only work on systems that allow forking (Sorry windows users). Future updates will allow for socketing.

Value

`output`	The list of character strings post-pretreatment

Author(s)

Travis Barton

Examples

```
## Not run:  # for some reason it takes longer than 5 seconds on CRAN's computers
test_vec = c('This is a test', 'Ahoy!', 'my battle-ship is on... b6!')
res = Pretreatment(test_vec)
print(res)

## End(Not run)
```

Random_Brains *Random Brains: Neural Network Implementation of Random Forest*

Description

Creates a random forest style collection of neural networks for classification

Usage

```
Random_Brains(data, y, x_test,
variables = ceiling(ncol(data)/10),
brains = floor(sqrt(ncol(data))),
hiddens = c(3, 4))
```

Arguments

`data` The data that holds the predictors ONLY.

`y` The responce variable

`x_test` The testing predictors

`variables` The number of predictors to select for each brain in 'data'. The default is one tenth of the number of columns in 'data'.

`brains` The number of neural networks to create. The default is the square root of the number of columns in 'data'.

`hiddens` The is a vector with length equal to the desired number of hidden layers. Each entry in the vector corresponds to the number of nodes in that layer. The default is c(3, 4) which is a two layer network with 3 and 4 nodes in the layers respectively.

Details

This function is meant to mirror the classic random forest function exctly. The only difference being that it uses shallow neural networks to build the forest instead of decision trees.

Value

`predictions` The predictions for x_test.

`num_brains` The number of neural networks used to decide the predictions.

`predictors_per_brain` The number of variabled used for the neural networks used to decide the predictions.

`hidden_layers` The vector describing the number of layers, as well as how many there were.

`preds_per_brain` This matrix describes which columns where selected by each brain. Each row is a new brain. each column describes the index of the column used.

`raw_results` The matrix of raw predictions from the brains. Each row is the cummulative predictions of all the brains. Which prediciton won by majority vote can be seen in 'predictions

Note

The neural networks are created using the neuralnet package!

Author(s)

Travis Barton

Examples

```
dat = Cross_val_maker(iris, .2)

train = dat$Train
test = dat$Test

Final_Test = Random_Brains(train[,-5],
  train$Species, as.matrix(test[,-5]),
  variables = 3, brains = 2)
table(Final_Test$predictions, as.numeric(test$Species))
```

Sentence_Vector *Function for extracting the sentence vector from an embeddings matrix.*

Description

Function for extracting the sentence vector from an embeddings matrix in a fast and convenient manner.

Usage

```
Sentence_Vector(Sentence, emb_matrix, dimension, stopwords)
```

Arguments

Sentence	The sentence to find the vector of.
emb_matrix	The embeddings matrix to search.
dimension	The dimension of the vector to return.
stopwords	Words that should not be included in the averaging proccess.

Details

The function splits the sentence into words, eliminates all stopwords, finds the vectors of each word, then averages the word vectors into a sentence vector.

Value

The sentence vector from an embeddings matrix.

Author(s)

Travis Barton

Examples

```
emb = data.frame(matrix(c(1, 2, 3, 4, 5, 5,
4, 3, 2, 1, 1, 5, 3, 2, 4), nrow = 3),
row.names = c('sentence', 'in', 'question'))

Sentence_Vector(c('this is the sentence in question'), emb, 5, c('this', 'is', 'the'))
```

Stopword_Maker *For the finding of the $N$ most populous words in a corpus.*

Description

This function finds the $N$ most used words in a corpus. This is done to identify stop words to better prune data sets before training.

Usage

```
Stopword_Maker(titles, cutoff = 20)
```

Arguments

`titles`	The documents in which the most populous words are sought.
`cutoff`	The number of $N$ top most used words to keep as stop words.

Value

`output`	A vector of the $N$ most populous words.

Author(s)

Travis Barton

Examples

```
test_set = c('this is a testset', 'I am searching for a list of words',
'I like turtles',
'A rocket would be a fast way of getting to work, but I do not think it is very practical')
res = Stopword_Maker(test_set, 4)
print(res)
```

`Table_percent` *Table Percent*

Description

Finds the acc of square tables.

Usage

```
Table_percent(in_table)
```

Arguments

`in_table`	a confusion matrix

Details

The table must be square

Note

make sure its square.

Author(s)

Travis Barton

Examples

```
true <- rep(1:10, 10)
test <- rep(1:10, 10)
test[c(2, 22, 33, 89)] = 1
Table_percent(table(true, test))
```

`Vector_Puller` *Function for extacting word vectors from embeddings.*

Description

Function for extacting word vectors from embeddings. This function is an internal function for 'Sentence_Puller'. It averages the word vectors and returns the average of these vectors.

Usage

```
Vector_Puller(words, emb_matrix, dimension)
```

Arguments

`words` The word to be extracted.

`emb_matrix` The embeddings matrix. It must be a data frame.

`dimension` The Dimension of the embeddings to extract. They do not have to match that of the matrix, but they cannot exceed its maximum column count.

Details

This is a simple and fast internal function.

Value

The vector that corresponds to the average of the word vectors.

Author(s)

Travis Barton

Examples

```
# This is an example emb_matrix

emb = data.frame(matrix(c(1, 2, 3, 4, 5, 5, 4, 3, 2, 1), nrow = 2), row.names = c('cow', 'moo'))

Vector_Puller(c('cow', 'moo'), emb, 5)
```

Index