Package 'RandPro'

January 20, 2025

Type Package

Title Random Projection with Classification

Version 0.2.2

Author Aghila G, Siddharth R

Maintainer Siddharth R <r.siddharthcse@gmail.com>

Description Performs random projection using Johnson-Lindenstrauss (JL) Lemma (see William B.Johnson and Joram Lindenstrauss (1984) <doi:10.1090/conm/026/737400>). Random Projection is a dimension reduction technique, where the data in the high dimensional space is projected into the low dimensional space using JL transform. The original high dimensional data matrix is multiplied with the low dimensional projection matrix which results in reduced matrix. The projection matrix can be generated using the projection function that is independent to the original data. Then finally apply the classification task on the projected data.

License GPL (>= 2)

Depends caret

Imports stats, e1071

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.1.1

Repository CRAN

NeedsCompilation no

Date/Publication 2020-07-19 14:50:03 UTC

Contents

classify 2
dimension 4
form_matrix 5

Index **7**

`classify` *Classification Function*

Description

The classify() function allows the user to combine the task of random projection based dimension reduction and classification within a single function. The dimension of the training data and test data was reduced by the value returned from the dimension() method. Then the projection matrix was generated using form_matrix() function based on the input paramater "projection".Then the training data and test data was projected into the low dimensional space by multiplying with the projection matrix. At last the reduced matrix was given to the classifier. The confusion matrix is the output of the classifier where we can calculate the performance of the classifier.

Usage

```
classify(
  train_data,
  test_data,
  train_label,
  test_label,
  eps = 0.1,
  projection = "gaussian",
  classifier = "knn"
)
```

Arguments

`train_data`	- Training data of either matrix or data frame
`test_data`	- Test data of either matrix or data frame
`train_label`	- Training label of either vector or data frame
`test_label`	- Test label of either vector or data frame
`eps`	- Epsilon with default 0.1
`projection`	- projection function with default "gaussian"
`classifier`	- classifier with default "knn"

Details

The parameters train_data,test_data,train_label and test_label are mandatory arguments. The eps is the error tolerance paramater. The value of eps must be $0.0 < eps < 1.0$. The default value of eps is 0.1 that means 10 percentage of error is acceptable during projection. The supported projection functions are gaussian, probability, li, and achlioptas.The default projection method is "gaussian". The complete detail of the projection function is given in form_matrix() function. The final argument "classifier" in the function defines the classifier to train the model. The supported classifier for classification task are

"knn" - k-nearest neighbor classification

"svmlinear" - Support Vector Machine

"nb" - Naive Bayes Classifier

Value

Confusion Matrix

Author(s)

Aghila G

Siddharth R

References

[1] Cannings, T. I. and Samworth, R. J. "Random projection ensemble classification(2015)".

[2] Ella Bingham and Heikki Mannila, "Random projection in dimensionality reduction: Applications to image and text data(2001)".

Examples

```
# Load Library
library(RandPro)

#Load Iris Data
data("iris")

#Split the data into training set and test set of 75:25 ratio.
set.seed(101)
sample <- sample.int(n = nrow(iris), size = floor(.75*nrow(iris)), replace = FALSE)
trainn <- iris[sample, ]
testt  <- iris[-sample,]

#Extract the train label and test label
trainl <- trainn$Species
testl <- testt$Species
typeof(trainl)

#Remove the label from training set and test set
trainn <- trainn[,1:4]
testt <- testt[,1:4]

#classify the Iris data with default K-NN Classifier.
res <- classify(trainn,testt,trainl,testl)
res
```

`dimension` *Function to determine the required number of dimension for generating the projection matrix*

Description

Johnson-Lindenstrauss (JL) lemma is the heart of random projection. The lemma states that a small set of points in a high-dimensional space can be embedded into low dimensional space in such a way that distances between the points are nearly preserved. The lemma has been used in dimensionality reduction, compressed sensing, manifold learning and graph embedding. The epsilon is the error tolerant parameter and it is inversely proportional to the accuracy of the result. The higher error tolerant level decreases the number of dimension and also the computation complexity with the marginal loss of accuracy.

Usage

```
dimension(sample, epsilon = 0.1)
```

Arguments

`sample`	- number of samples
`epsilon`	- error tolerance level with default value 0.1

Details

The function dimension() is used to find the minimum dimension required to project the data from high dimensional space to low dimensional space. The number of sample and error tolerant level has been passed as an input argument to the function dimension() . It will return the size of the random subspace to guarantee a bounded distortion introduced by the random projection.

Value

minimum number of dimension required to maintain the pair wise distance between any two points with the controlled amount of error(eps)

Author(s)

Aghila G

Siddharth R

References

[1] William B.Johnson, Joram Lindenstrauss, "Extension of Lipschitz mappings into a Hilbert space (1984)"

[2] Sanjoy Dasgupta , Anupam Gupta "An elementary proof of a theorem of Johnson and Lindenstrauss (2003)"

See Also

Johnson-Lindenstrauss Elementary Proof

Examples

```
#load library
library(RandPro)

#Calculate minimum dimension using eps =0.5 for 1000000 sample
y <- dimension(1000000,0.5)

#Calculating minimum dimension using different epsilon value for 1000000 sample
d <-  c(0.5,0.1)
x<- dimension(103260,d)
```

`form_matrix` *Forms the Projection Matrix*

Description

The projection function is used to generate the random projection matrix. It will form either dense or spare projection matrix. The Package supports 4 projection functions namely gaussian, probability, achlioptas and li. The number of rows and columns of the input sample is passed with the boolean value JLT. If JLT is set to TRUE, the dimension of the input data is reduced to the value returned by dimension() method. For Dense Matrix - "gaussian" method For Sparse Matrix - "probability, achlioptas and li" method.

Usage

```
form_matrix(rows, cols, JLT, eps = 0.1, projection = "gaussian")
```

Arguments

`rows`	- number of rows
`cols`	- number of columns
`JLT`	- Boolean to set JL transform (TRUE or FALSE)
`eps`	- error tolerance level with default value 0.1
`projection`	- projection function with default value "gaussian"

Details

The 4 projection functions are

1."gaussian" - The default projection function is "gaussian". In probability theory, Gaussian distribution is also called as normal distribution. It is a continuous probability distribution used to represent real-valued random variables. The elements in the random matrix are drawn from N(0,1/k), N is a Natural number and k value calculated based on JL - Lemma using dimension() function.

2. "probability" - In this method, the matrix was generated using the equal probability distribution with the elements [-1, 1].

3. "achlioptas" - Achlioptas matrix is easy to generate and also the 2/3rd of the matrix was filled with zero which makes it as more sparse and cut-off the 2/3rd computation.

4. "li" - This method generalizes the achlioptas method and generate very sparse random matrix to improve the computational speed up of random projection.

When comparing to gaussian function, the other projection functions creates sparse matrix by filling with zero's or one's to reduce the computation even more.

Value

Projection Matrix

Author(s)

Aghila G

Siddharth R

References

[1] N.I.R. Ailon and B.Chazelle, "The Fast Johnson Lindenstrauss Transform and Approximate Nearest Neighbors(2009)"

[2] Ping Li, Trevor J. Hastie, and Kenneth W. Church, "Very sparse random projections(2006)".

[3] D. Achlioptas, "Database-friendly random projections(2002)"

Examples

```
# Load Library
library(RandPro)

# Default Gaussian projection matrix without JL transform
mat <- form_matrix(600,1000,FALSE)

# Default Gaussian projection matrix with JL transform of 50% Error tolerance
mat <- form_matrix(300,100000,TRUE,0.5)

# Projection matrix with probability distribution of 50% Error tolerance
mat <- form_matrix(250,1000000,TRUE,0.5,"probability")

# Projection matrix with li distribution of 50% Error tolerance
mat <- form_matrix(250,1000000,TRUE,0.5,"li")

# Projection matrix with achlioptas distribution of 50% Error tolerance
mat <- form_matrix(250,1000000,TRUE,0.5,"achlioptas")
```

Index

∗ **Achlioptas**
    form_matrix, 5
∗ **Dimension_Reduction**
    dimension, 4
∗ **Distribution**
    form_matrix, 5
∗ **Gaussian**
    form_matrix, 5
∗ **Johnson-Lindenstrauss_Lemma**
    dimension, 4
∗ **Li**
    form_matrix, 5
∗ **Probability**
    form_matrix, 5
∗ **Projection_matrix**
    form_matrix, 5
∗ **Random_projection**
    dimension, 4
∗ **classification**
    classify, 2
∗ **confusion_matrix**
    classify, 2
∗ **feature_extraction**
    classify, 2
∗ **k-nn**
    classify, 2
∗ **sparse_matrix**
    form_matrix, 5
∗ **svm**
    classify, 2

classify, 2

dimension, 4

form_matrix, 5