Package 'MultivariateRandomForest'

July 21, 2025

Type Package

Title Models Multivariate Cases Using Random Forests

Version 1.1.5

Date 2017-04-05

Author Raziur Rahman

Maintainer Raziur Rahman <razeeebuet@gmail.com>

Description Models and predicts multiple output features in single random forest considering the linear relation among the output features, see details in Rahman et al (2017)<doi:10.1093/bioinformatics/btw765>.

License GPL (>= 2)

RoxygenNote 6.0.1

Depends R (>= 2.10)

Imports Rcpp, bootstrap, stats

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-05-01 10:20:31 UTC

Contents

`build_forest_predict` *Prediction using Random Forest or Multivariate Random Forest*

Description

Builds Model of Random Forest or Multivariate Random Forest (when the number of output features > 1) using training samples and generates the prediction of testing samples using the inferred model.

Usage

```
build_forest_predict(trainX, trainY, n_tree, m_feature, min_leaf, testX)
```

Arguments

`trainX`	Input Feature matrix of M x N, M is the number of training samples and N is the number of input features
`trainY`	Output Response matrix of M x T, M is the number of training samples and T is the number of ouput features
`n_tree`	Number of trees in the forest, which must be positive integer
`m_feature`	Number of randomly selected features considered for a split in each regression tree node, which must be positive integer and less than N (number of input features)
`min_leaf`	Minimum number of samples in the leaf node. If a node has less than or equal to min_leaf samples, then there will be no splitting in that node and this node will be considered as a leaf node. Valid input is positive integer, which is less than or equal to M (number of training samples)
`testX`	Testing samples of size Q x N, where Q is the number of testing samples and N is the number of features (Same number of features as training samples)

Details

Random Forest regression refers to ensembles of regression trees where a set of n_tree un-pruned regression trees are generated based on bootstrap sampling from the original training data. For each node, the optimal feature for node splitting is selected from a random set of m_feature from the total N features. The selection of the feature for node splitting from a random set of features decreases the correlation between different trees and thus the average prediction of multiple regression trees is expected to have lower variance than individual regression trees. Larger m_feature can improve the predictive capability of individual trees but can also increase the correlation between trees and void any gains from averaging multiple predictions. The bootstrap resampling of the data for training each tree also increases the variation between the trees.

In a node with training predictor features (X) and output feature vectors (Y), node splitting is done with the aim of selecting a feature from a random set of m_feature and threshold z to partition the node into two child nodes, left node (with samples < z) and right node (with samples >=z). In multivariate trees (MRF) node cost is measured as the sum of squares of the Mahalanobis distance where as in univariate trees (RF) node cost is measured as the Euclidean distance.

After the Model of the forest is built using training Input features (trainX) and output feature matrix (trainY), the Model is used to generate the prediction of output features (testY) for the testing samples (testX).

Value

Prediction result of the Testing samples

References

[Random Forest] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

[Multivariate Random Forest] Segal, Mark, and Yuanyuan Xiao. "Multivariate random forests." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1.1 (2011): 80-87.

Examples

```
library(MultivariateRandomForest)
#Input and Output Feature Matrix of random data (created using runif)
trainX=matrix(runif(50*100),50,100)
trainY=matrix(runif(50*5),50,5)
n_tree=2
m_feature=5
min_leaf=5
testX=matrix(runif(10*100),10,100)
#Prediction size is 10 x 5, where 10 is the number
#of testing samples and 5 is the number of output features
Prediction=build_forest_predict(trainX, trainY, n_tree, m_feature, min_leaf, testX)
```

---

build_single_tree *Model of a single tree of Random Forest or Multivariate Random Forest*

Description

Build a Univariate Regression Tree (for generation of Random Forest (RF)) or Multivariate Regression Tree (for generation of Multivariate Random Forest (MRF)) using the training samples, which is used for the prediction of testing samples.

Usage

```
build_single_tree(X, Y, m_feature, min_leaf, Inv_Cov_Y, Command)
```

Arguments

X	Input Feature matrix of M x N, M is the number of training samples and N is the number of input features
Y	Output Feature matrix of M x T, M is the number of training samples and T is the number of ouput features
m_feature	Number of randomly selected features considered for a split in each regression tree node, which must be positive integer and less than N (number of input features)
min_leaf	Minimum number of samples in the leaf node, which must be positive integer and less than or equal to M (number of training samples)
Inv_Cov_Y	Inverse of Covariance matrix of Output Response matrix for MRF(Input [0 0;0 0] for RF)
Command	1 for univariate Regression Tree (corresponding to RF) and 2 for Multivariate Regression Tree (corresponding to MRF)

Details

The regression tree structure is represented as a list of lists. For a non-leaf node, it contains the splitting criteria (feature for split and threshold) and for a leaf node, it contains the output responses for the samples contained in the leaf node.

Value

Model of a single regression tree (Univariate or Multivariate Regression Tree). An example of the list of the non-leaf node:

`Flag for determining whether the node is leaf node or branch node. 0 means branch node and 1 means leaf node.`
1

`Index of samples for the left node`
int [1:34] 1 2 4 5 ...

`Index of samples for the right node`
int [1:16] 3 6 9 ...

`Feature for split`
int 34

`Threshold values for split, average them`
num [1:3] 0.655 0.526 0.785

`List number for the left and right nodes`
num [1:2] 2 3

An example of the list of the leaf node:

`Output responses`
num[1:4,1:5] 0.0724 0.1809 0.0699 ...

CrossValidation — *Generate training and testing samples for cross validation*

Description

Generates Cross Validation Input Matrices and Output Vectors for training and testing, where number of folds in cross validation is user defined.

Usage

```
CrossValidation(X, Y, F)
```

Arguments

X	M x N Input matrix, M is the number of samples and N is the number of features
Y	output responses as column vector
F	Number of Folds

Value

List with the following components:

TrainingData	List of matrices where each matrix contains a fold of Cross Validation Training Data, where the number of matrices is equal to F
TestingData	List of matrices where each matrix contains a fold of Cross Validation Testing Data, where the number of matrices is equal to F
OutputTrain	List of matrices where each matrix contains a fold of Cross Validation Training Output Feature Data, where the number of matrices is equal to F
OutputTest	List of matrices where each matrix contains a fold of Cross Validation Testing Output Feature Data, where the number of matrices is equal to F
FoldedIndex	Index of Different Folds. (e.g., for Sample Index 1:6 and 3 fold, FoldedIndex are [1 2 3 4], [1 2 5 6], [3 4 5 6])

Imputation — *Imputation of a numerical vector*

Description

Imputes the values of the vector that are NaN

Usage

```
Imputation(XX)
```

Arguments

XX	a vector of size N x 1

Details

If a value is missing, it will be replaced by an imputed value that is an average of previous and next value. If previous or next value is also missing, the closest value is used as the imputed value.

Value

Imputed vector of size N x 1

Node_cost *Information Gain*

Description

Compute the cost function of a tree node

Usage

```
Node_cost(y, Inv_Cov_Y, Command)
```

Arguments

y	Output Features for the samples of the node
Inv_Cov_Y	Inverse of Covariance matrix of Output Response matrix for MRF(Input [0 0;0 0] for RF)
Command	1 for univariate Regression Tree (corresponding to RF) and 2 for Multivariate Regression Tree (corresponding to MRF)

Details

In multivariate trees (MRF) node cost is measured as the sum of squares of the Mahalanobis distance to capture the correlations in the data whereas in univariate trees node cost is measured as the sum of Euclidean distance square. Mahalanobis Distance captures the distance of the sample point from the mean of the node along the principal component axes.

Value

cost or entropy of samples in a node of a tree

References

Segal, Mark, and Yuanyuan Xiao. "Multivariate random forests." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1.1 (2011): 80-87.

Examples

```
library(MultivariateRandomForest)
y=matrix(runif(10*2),10,2)
Inv_Cov_Y=solve(cov(y))
Command=2
#Command=2 for MRF and 1 for RF
#This function calculates information gain of a node
Cost=Node_cost(y,Inv_Cov_Y,Command)
```

`predicting` *Prediction of testing sample in a node*

Description

Provides the value of a testing sample in a node which refers to which child node it will go using the splitting criteria of the tree node or prediction results if the node is a leaf.

Usage

```
predicting(Single_Model, i, X_test, Variable_number)
```

Arguments

`Single_Model`	Model of a particular tree
`i`	Number of split. Used as an index, which indicates where in the list the splitting criteria of this split has been stored.
`X_test`	Testing samples of size Q x N, Q is the number of testing samples and N is the number of features (same order and size used as training)
`Variable_number`	Number of Output Features

Details

The function considers the output at a particular node. If the node is a leaf, the average of output responses is returned as prediction result. For a non-leaf node, the direction of left or right node is decided based on the node threshold and splitting feature value.

Value

Prediction result of a testing samples in a node

single_tree_prediction *Prediction of Testing Samples for single tree*

Description

Predicts the output responses of testing samples based on the input regression tree

Usage

```
single_tree_prediction(Single_Model, X_test, Variable_number)
```

Arguments

Single_Model	Random Forest or Multivariate Random Forest Model of a particular tree
X_test	Testing samples of size Q x N, Q is the number of testing samples and N is the number of features (same order and size used as training)
Variable_number	Number of Output Features

Details

A regression tree model contains splitting criteria for all the splits in the tree and output responses of training samples in the leaf nodes. A testing sample using these criteria will reach a leaf node and the average of the Output response vectors in the leaf node is considered as the prediction of the testing sample.

Value

Prediction result of the Testing samples for a particular tree

splitt2 *Split of the Parent node*

Description

Split of the training samples of the parent node into the child nodes based on the feature and threshold that produces the minimum cost

Usage

```
splitt2(X, Y, m_feature, Index, Inv_Cov_Y, Command, ff)
```

Arguments

X	Input Training matrix of size M x N, M is the number of training samples and N is the number of features
Y	Output Training response of size M x T, M is the number of samples and T is the number of output responses
m_feature	Number of randomly selected features considered for a split in each regression tree node.
Index	Index of training samples
Inv_Cov_Y	Inverse of Covariance matrix of Output Response matrix for MRF (Input [0 0; 0 0] for RF)
Command	1 for univariate Regression Tree (corresponding to RF) and 2 for Multivariate Regression Tree (corresponding to MRF)
ff	Vector of m_feature from all features of X. This varies with each split

Details

At each node of a regression a tree, a fixed number of features (m_feature) are selected randomly to be considered for generating the split. Node cost for all selected features along with possible n-1 thresholds for n samples are considered to select the feature and threshold with minimum cost.

Value

List with the following components:

index_left	Index of the samples that are in the left node after splitting
index_right	Index of the samples that are in the right node after splitting
which_feature	The number of the feature that produces the minimum splitting cost
threshold_feature	The threshold value for the node split. A feature value less than or equal to the threshold will go to the left node and it will go to the right node otherwise.

Examples

```
library(MultivariateRandomForest)
X=matrix(runif(20*100),20,100)
Y=matrix(runif(20*3),20,3)
m_feature=5
Index=1:20
Inv_Cov_Y=solve(cov(Y))
ff2 = ncol(X) # number of features
ff =sort(sample(ff2, m_feature))
Command=2#MRF, as number of output feature is greater than 1
Split_criteria=splitt2(X,Y,m_feature,Index,Inv_Cov_Y,Command,ff)
```

split_node *Splitting Criteria of all the nodes of the tree*

Description

Stores the Splitting criteria of all the nodes of a tree in a list

Usage

```
split_node(X, Y, m_feature, Index, i, model, min_leaf, Inv_Cov_Y, Command)
```

Arguments

X	Input Training matrix of size M x N, M is the number of training samples and N is the number of features
Y	Output Training response of size M x T, M is the number of samples and T is the number of output responses
m_feature	Number of randomly selected features considered for a split in each regression tree node
Index	Index of training samples
i	Number of split. Used as an index, which indicates where in the list the splitting criteria of this split will be stored.
model	A list of lists with the spliting criteria of all the node splits. In each iteration, a new list is included with the spliting criteria of the new split of a node.
min_leaf	Minimum number of samples in the leaf node. If a node has less than or, equal to min_leaf samples, then there will be no splitting in that node and the node is a leaf node. Valid input is a positive integer and less than or equal to M (number of training samples)
Inv_Cov_Y	Inverse of Covariance matrix of Output Response matrix for MRF(Give Zero for RF)
Command	1 for univariate Regression Tree (corresponding to RF) and 2 for Multivariate Regression Tree (corresponding to MRF)

Details

This function calculates the splitting criteria of a node and stores the information in a list format. If the node is a parent node, then indices of left and right nodes and feature number and threshold value of the feature for the split are stored. If the node is a leaf, the output feature matrix of the samples for the node are stored as a list.

Value

Model: A list of lists with the splitting criteria of all the split of the nodes. In each iteration, the Model is updated with a new list that includes the splitting criteria of the new split of a node.

variable_importance_measure *Calculates variable Importance of a Regression Tree Model*

Description

Number of times a variable has been picked in the branch nodes of a (single) regression tree.

Usage

```
variable_importance_measure(Model_VIM,NumVariable)
```

Arguments

Model_VIM	Regression Tree model in which the variable importance is measured
NumVariable	Number of variables in the training or testing matrix

Details

In time of calculating node cost of a tree of a random forest, a user defined number of variables are randomly picked. Among this, the best variable is chosen for the node using the node cost. While an important variable for a model will always come out as the best. This function calculates the number of times a variable has been picked in the regression tree. It has been done by checking which variables are picked, how many times, in the branch nodes of the model.

Value

Vector of size (1 x NumVariable), showing the number of repetition of variables (serially) in the branch nodes of the model.

Examples

```
library(MultivariateRandomForest)
trainX=matrix(runif(50*100),50,100)
trainY=matrix(runif(50*5),50,5)
n_tree=2
m_feature=5
min_leaf=5
testX=matrix(runif(10*100),10,100)

theta <- function(trainX){trainX}
results <- bootstrap::bootstrap(1:nrow(trainX),n_tree,theta)
b=results$thetastar

Variable_number=ncol(trainY)
if (Variable_number>1){
  Command=2
}else if(Variable_number==1){
  Command=1
```

```
}
NumVariable=ncol(trainX)
NumRepeatation=matrix(rep(0,n_tree*NumVariable),nrow=n_tree)

for (i in 1:n_tree){
  Single_Model=NULL
  X=trainX[ b[ ,i],  ]
  Y=matrix(trainY[ b[ ,i],  ],ncol=Variable_number)
  Inv_Cov_Y = solve(cov(Y)) # calculate the V inverse
  if (Command==1){
    Inv_Cov_Y=matrix(rep(0,4),ncol=2)
  }
  Single_Model=build_single_tree(X, Y, m_feature, min_leaf,Inv_Cov_Y,Command)
  NumRepeatation[i,]=variable_importance_measure(Single_Model,NumVariable)
}
```

Index