Package: IntegratedMRF (via r-universe)

September 8, 2024

Type Package

Title Integrated Prediction using Uni-Variate and Multivariate Random Forests

Version 1.1.9

Date 2018-07-05

Author Raziur Rahman, Ranadip Pal

Maintainer Raziur Rahman <razeeebuet@gmail.com>

Description An implementation of a framework for drug sensitivity prediction from various genetic characterizations using ensemble approaches. Random Forests or Multivariate Random Forest predictive models can be generated from each genetic characterization that are then combined using a Least Square Regression approach. It also provides options for the use of different error estimation approaches of Leave-one-out, Bootstrap, N-fold cross validation and 0.632+Bootstrap along with generation of prediction confidence interval using Jackknife-after-Bootstrap approach.

License GPL-3

RoxygenNote 6.0.1

Depends R (>= 2.10)

Imports Rcpp (>= 0.12.4), bootstrap, ggplot2, utils, stats, limSolve, MultivariateRandomForest

LinkingTo Rcpp

NeedsCompilation yes

Date/Publication 2018-07-05 20:30:03 UTC

Repository https://razrahman.r-universe.dev

RemoteUrl https://github.com/cran/IntegratedMRF

RemoteRef HEAD

RemoteSha d049a5140ee3d6f87e1c5daf809ae28cbf73ee8f

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(IntegratedMRF)
#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 node status; leaf node (1) or branch node (0)`
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 ...

`Combination` *Weights for combination of predictions from different data subtypes using Least Square Regression based on various error estimation techniques*

Description

Calculates combination weights for different subtypes of dataset combinations to generate integrated Random Forest (RF) or Multivariate Random Forest (MRF) model based on different error estimation models such as Bootstrap, 0.632+ Bootstrap, N-fold cross validation or Leave one out.

Usage

```
Combination(finalX, finalY_train, Cell, finalY_train_cell, n_tree, m_feature,
  min_leaf, Confidence_Level)
```

Arguments

`finalX`
: List of Matrices where each matrix represent a specific data subtype (such as genomic characterizations for drug sensitivity prediction). Each subtype can have different types of features. For example, if there are three subtypes containing 100, 200 and 250 features respectively, finalX will be a list containing 3 matrices of sizes M x 100, M x 200 and M x 250 where M is the number of Samples.

`finalY_train`
: A M x T matrix of output features for training samples, where M is number of samples and T is the number of output features. The dataset is assumed to contain no missing values. If there are missing values, an imputation method should be applied before using the function. A function 'Imputation' is included within the package.

`Cell`
: It contains a list of samples (the samples can be represented either numerically by indices or by names) for each data subtype. For the example of 3 data subtypes, it will be a list containing 3 arrays where each array contains the sample information for each data subtype.

`finalY_train_cell`
: Sample names of output features for training samples

`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, Valid Input is a positive integer, which is less than N (which is equal to number of input features for the smallest genomic characterization)

`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)

`Confidence_Level` Confidence level for calculation of confidence interval (User Defined), which must be between 0 and 100

Details

The function takes all the subtypes of dataset in matrix format and its corresponding sample information. For calculation purpose, we have considered the data of the samples that are common in all the subtypes and output training responses. For example, consider a dataset of 3 sub-types with different number of samples and features, with indices of samples in subtype 1, 2, 3 and output feature matrix is 1:10, 3:15, 5:16 and 5:11 respectively. Thus, features of sample index 5:10 (common to all subtypes and output feature matrix) of all subtypes and output feature matrix will be selected and considered for all calculations.

For M x N dataset, N number of bootstrap sampling sets are considered. For each bootstrap sampling set and each subtype, a Random Forest (RF) or, Multivariate Random Forest (MRF) model is generated, which is used for calculating the prediction performance for out-of-bag samples. The prediction performance for each subtype of the dataset is based on the averaging over different bootstrap training sets. The combination weights (regression coefficients) for each combination of subtypes are generated using least Square Regression from the individual subtype predictions and used later to calculate mean absolute error, mean square error and correlation coefficient between predicted and actual values.

For N-fold cross validation error estimation with M cell lines, N models are generated for each subtype of dataset, where for each partition (M/N)*(N-1) cell lines are used for training and the remaining cell lines are used to estimate errors and combination weights for different data subtype combinations.

In 0.632 Bootstrap error estimation, bootstrap and re-substitution error estimates are combined based on 0.632xBootstrap Error + 0.368xRe-substitution Error. While 0.632+ Bootstrap error estimation considers the overfitting of re-substitution error with no information error rate γ. An estimate of γ is obtained by permuting the responses $y[i]$ and predictors $x[j]$.

$$\gamma = sum(sum(error(x[j], y[i]), j = 1, m), i = 1, m)/m^2$$

The relative overfitting rate is defined as $R = (BootstrapError - ResubstitutionError)/(\gamma - ResubstitutionError)$ and weight distribution between bootstrap error and Re-substitution Error is defined as $w = 0.632/(1 - 0.368 * R)$. So, 0.632+ Bootstrap error is equal to $(1 - w) * BootstrapError + w * ResubstitutionError$. These prediction results are then used to compute the errors and combination weights for different data subtype combinations.

Confidence Interval has been calculated using Jackkniffe-After-Bootstrap Approach and prediction result of bootstrap error estimation.

For leave-one-out error estimation using M cell lines, M models are generated for each subtype of dataset, which are then used to calculate the errors and combination weights for different data subtype combinations.

Value

List with the following components:

`BSP_coeff` Combination weights using Bootstrap Error Estimation Model, where index is in list format. If the number of genomic characterizations or subtypes of dataset is 5, there will be 2^5-1=31 list of weights

`Nfold_coeff` Combination weights using N fold cross validation Error Estimation Model, where index is in list format. If the number of genomic characterizations or subtypes of dataset is 5, there will be 2^5-1=31 list of weights

`BSP632plus_coeff` Combination weights using 0.632+ Bootstrap Error Estimation Model, where index is in list format. If the number of genomic characterizations or subtypes of dataset is 5, there will be 2^5-1=31 list of weights

`LOO_coeff` Combination weights using Leave-One-Out Error Estimation Model, where index is in list format. If the number of genomic characterizations or subtypes of dataset is 5, there will be 2^5-1=31 list of weights

`Error` Matrix of Mean Absolute Error, Mean Square Error and correlation between actual and predicted responses for integrated model based on Bootstrap, N fold cross validation, 0.632+ Bootstrap and Leave-one-out error estimation sampling techniques for the integrated model containing all the data subtypes

`Confidence Interval` Low and High confidence interval for a user defined confidence level for the drug using Jackknife-After-Bootstrap Approach in a list

`BSP_error_all_mae` Bootstrap Mean Absolute Errors (MAE) for all combinations of the dataset subtypes. Size C x R, where C is the number of combinations and R is the number of output responses. C is in decreasing order, which means first value is combination of all subtypes and next ones are in decreasing order. For example, if a dataset has 3 subtypes, then C is equal to 2^3-1=7. The ordering of C is the combination of subtypes [1 2 3], [1 2], [1 3], [2 3], [1], [2], [3]

`Nfold_error_all_mae` N fold cross validation Mean Absolute Errors (MAE) for all combinations of the dataset subtypes. Size C x R, where C is the number of combinations and R is the number of output responses. C is in decreasing order, which means first value is combination of all subtypes and next ones are in decreasing order. For example, if a dataset has 3 subtypes, then C is equal to 2^3-1=7. The ordering of C is the combination of subtypes [1 2 3], [1 2], [1 3], [2 3], [1], [2], [3]

`BSP632plus_error_all_mae` 0.632+ Bootstrap Mean Absolute Errors (MAE) for all combinations of the dataset subtypes. Size C x R, where C is the number of combinations and R is the number of output responses. C is in decreasing order, which means first value is combination of all subtypes and next ones are in decreasing order. For example, if a dataset has 3 subtypes, then C is equal to 2^3-1=7. The ordering of C is the combination of subtypes [1 2 3], [1 2], [1 3], [2 3], [1], [2], [3]

`LOO_error_all_mae` Leave One Out Mean Absolute Errors (MAE) for all combinations of the dataset subtypes. Size C x R, where C is the number of combinations and R is the

number of output responses. C is in decreasing order, which means first value is combination of all subtypes and next ones are in decreasing order. For example, if a dataset has 3 subtypes, then C is equal to 2^3-1=7. The ordering of C is the combination of subtypes [1 2 3], [1 2], [1 3], [2 3], [1], [2], [3]

The function also returns figures of different error estimations in .tiff format

References

Wan, Qian, and Ranadip Pal. "An ensemble based top performing approach for NCI-DREAM drug sensitivity prediction challenge." PloS one 9.6 (2014): e101183.

Rahman, Raziur, John Otridge, and Ranadip Pal. "IntegratedMRF: random forest-based framework for integrating prediction from different data types." Bioinformatics (Oxford, England) (2017).

Efron, Bradley, and Robert Tibshirani. "Improvements on cross-validation: the 632+ bootstrap method." Journal of the American Statistical Association 92.438 (1997): 548-560.

Examples

```
library(IntegratedMRF)
data(Dream_Dataset)
Tree=1
Feature=1
Leaf=5
Confidence=80
finalX=Dream_Dataset[[1]]
Cell=Dream_Dataset[[2]]
Y_train_Dream=Dream_Dataset[[3]]
Y_train_cell=Dream_Dataset[[4]]
Y_test=Dream_Dataset[[5]]
Y_test_cell=Dream_Dataset[[6]]
Drug=c(1,2,3)
Y_train_Drug=matrix(Y_train_Dream[,Drug],ncol=length(Drug))
Result=Combination(finalX,Y_train_Drug,Cell,Y_train_cell,Tree,Feature,Leaf,Confidence)
```

`CombPredict` *Integrated Prediction of Testing samples using Combination Weights from integrated RF or MRF model*

Description

Generates Random Forest or Multivariate Random Forest model for each subtype of dataset and predicts testing samples using the generated models. Subsequently, the prediction for different subtypes of dataset are combined using the Combination weights generated from 'Combination' function.

Usage

```
CombPredict(finalX, finalY_train, Cell, finalY_train_cell, finalY_test_cell,
  n_tree, m_feature, min_leaf, Coeff)
```

Arguments

finalX	List of Matrices where each matrix represents a specific data subtype (such as genomic characterizations for drug sensitivity prediction). Each subtype can have different types of features. For example, if there are three subtypes containing 100, 200 and 250 features respectively, finalX will be a list containing 3 matrices of sizes M x 100, M x 200 and M x 250 where M is the number of Samples.
finalY_train	A M x T matrix of output features for training samples, where M is number of samples and T is the number of output features. The dataset is assumed to contain no missing values. If there are missing values, an imputation method should be applied before using the function. A function 'Imputation' is included within the package.
Cell	It contains a list of samples (the samples can be represented either numerically by indices or by names) for each data subtype. For the example of 3 data subtypes, it will be a list containing 3 arrays where each array contains the sample information for each data subtype.
finalY_train_cell	Cell lines of output features for training samples
finalY_test_cell	Cell lines of output features for testing samples
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
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)
Coeff	Combination Weights. The user can supply the weights based on either Bootstrap, Re-substitution, 0.632Bootstrap or Leave-one-out error estimation approaches.

Details

Input matrix and output response of training samples have been used to build Random Forest or Multivariate Random Forest model for each subtype of a dataset. These models are used to calculate prediction of testing samples for each subtype separately. Subsequently Combination Weights (different errors have different combination weights and the user should select the one to be used) are used to integrate the predictions from data subtypes. Note that the combination weights are linear regression coefficients generated using the training samples.

The specific set of combination weights to be used for testing samples will depend on the number of data subtypes available for the testing samples. Note that not all subtype information maybe available for all samples. As an example with three data subtypes, a testing sample with all subtype data available will use the combination weights corresponding to Serial [1 2 3] where if subtype 3 is not available, the function will using the combination weights corresponding to Serial [1 2].

Value

Final Prediction of testing samples based on provided testing sample names.

Examples

```
library(IntegratedMRF)
data(Dream_Dataset)
Tree=1
Feature=1
Leaf=10
Confidence=80
finalX=Dream_Dataset[[1]]
Cell=Dream_Dataset[[2]]
Y_train_Dream=Dream_Dataset[[3]]
Y_train_cell=Dream_Dataset[[4]]
Y_test=Dream_Dataset[[5]]
Y_test_cell=Dream_Dataset[[6]]
Drug=1
Y_train_Drug=matrix(Y_train_Dream[,Drug],ncol=length(Drug))
Result=Combination(finalX,Y_train_Drug,Cell,Y_train_cell,Tree,Feature,Leaf,Confidence)

CombPredict(finalX,Y_train_Drug,Cell,Y_train_cell,Y_test_cell,Tree,Feature,Leaf,Result[[1]])
```

CombPredictSpecific *Prediction for testing samples using specific combination weights from integrated RF or MRF model*

Description

Generates Random Forest (One Output Feature) or Multivariate Random Forest (More than One Output Feature) model for each subtype of dataset and predicts testing samples using these models. The predictions are combined using the specific combination weights provided by the user. For the input combination weights, the testing cell lines should have the subtype data corresponding to the non-zero weight subtypes.

Usage

```
CombPredictSpecific(finalX, finalY_train, Cell, finalY_train_cell,
  finalY_test_cell, n_tree, m_feature, min_leaf, Coeff)
```

Arguments

finalX	List of Matrices where each matrix represent a specific data subtype (such as genomic characterizations for drug sensitivity prediction). Each subtype can have different types of features. For example, if there are three subtypes containing 100, 200 and 250 features respectively, finalX will be a list containing 3 matrices of sizes M x 100, M x 200 and M x 250 where M is the number of Samples.
finalY_train	A M x T matrix of output features for training samples, where M is the number of samples and T is the number of output features. The dataset is assumed to contain no missing values. If there are missing values, an imputation method should be applied before using the function. A function 'Imputation' is included within the package.

Cell	It contains a list of samples (the samples can be represented either numerically by indices or by names) for each data subtype. For the example of 3 data subtypes, it will be a list containing 3 arrays where each array contains the sample information for each data subtype.
finalY_train_cell	Sample names of output features for training samples
finalY_test_cell	Sample names of output features for testing samples (All these testing samples must have features for each subtypes of dataset)
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 a positive integer
min_leaf	Minimum number of samples in the leaf node, which must be a positive integer less than or equal to M (number of training samples)
Coeff	Combination Weights (user defined or some combination weights generated using the 'Combination' function). The size must be C, which is equal to the number of subtypes of dataset given in finalX.

Details

Input feature matrix and output feature matrix have been used to generate Random Forest (One Output Feature) or Multivariate Random Forest (More than One Output Feature) model for each subtype of dataset separately. The prediction of testing samples using each subtype trained model is generated. The predictions are combined using the specific combination weights provided by the user. For the input combination weights, the testing cell lines should have the subtype data corresponding to the non-zero weight subtypes. For instance, if combination weights is [0.6 0.3 0 0.1], then the subtype 1, 2 and 4 needs to be present for the testing samples. Furthermore, all the features should be present for the required subtypes for the testing samples.

Value

Final Prediction of testing samples based on provided testing sample names

Examples

```
library(IntegratedMRF)
data(Dream_Dataset)
Tree=1
Feature=1
Leaf=10
Confidence=80
finalX=Dream_Dataset[[1]]
Cell=Dream_Dataset[[2]]
Y_train_Dream=Dream_Dataset[[3]]
Y_train_cell=Dream_Dataset[[4]]
Y_test=Dream_Dataset[[5]]
Y_test_cell=Dream_Dataset[[6]]
Drug=1
Y_train_Drug=matrix(Y_train_Dream[,Drug],ncol=length(Drug))
```

```
Result=Combination(finalX,Y_train_Drug,Cell,Y_train_cell,Tree,Feature,Leaf,Confidence)

CombPredictSpecific(finalX,Y_train_Drug,Cell,Y_train_cell,Y_test_cell,Tree,
        Feature,Leaf,runif(length(Cell)*1))
```

`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 response 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])

Dream_Dataset *NCI-Dream Drug Sensitivity Prediction Challenge Dataset*

Description

A demo dataset of different genomic characterizations and drug sensitivity selected from NCI-Dream Drug Sensitivity Prediction Challenge dataset.

Usage

```
Dream_Dataset
```

Format

A list of 6 variables containing genomic characterizations and drug sensitivity:

finalX_Dream List of 5 Matrices where the matrices represent different genomic characterizations of Gene Expression, Methylation, RNA sequencing, Reverse Phase Protein Array (RPPA) and Copy Number Variation (CNV). 1000 predictor features for each subtype is included to satisfy package size limitations.

Cell_line_Index_Dream List of Cell Line names for each genomic charcterization

finalY_train_Dream Drug Sensitivity of training samples (35) for 31 drugs provided for NCI-Dream Drug Sensitivity Prediction Challenge

finalY_train_cell_Dream Cell line names of the training samples

finalY_test_Dream Drug Sensitivity of testing samples (18) for 31 drugs provided for NCI-Dream Drug Sensitivity Prediction Challenge Dataset

finalY_test_cell_Dream Cell line names of the testing samples

Source

https://www.synapse.org/#!Synapse:syn2785778/wiki/70252

References

Costello, James C., et al. "A community effort to assess and improve drug sensitivity prediction algorithms." Nature biotechnology 32.12 (2014): 1202-1212.

`error_calculation` *Error calculation for integrated model*

Description

Combines Prediction from different data subtypes through Least Square Regression and computes Mean Absolute Error, Mean Square Error and Pearson Correlation Coefficient between Integrated Prediction and Original Output feature.

Usage

```
error_calculation(final_pred, final_actual)
```

Arguments

`final_pred`	A n x p matrix of predicted features, where n is the number of samples and p is the number of data subtypes with prediction
`final_actual`	A n x 1 vector of original output responses

Details

If final_pred is a vector, it refers to the prediction result for one subtype of dataset and this function will return Mean Absolute Error, Mean Square Error and Pearson Correlation Coefficient between predicted and Original Output response. If final_pred is a matrix containing prediction results for more than one subtype of dataset, Least Square Regression will be used to calculate the weights for combining the predictions and generate an integrated prediction of size n x 1. Subsequently, Mean Absolute Error, Mean Square Error and Pearson Correlation Coefficient between Integrated Prediction and Original Output responses are calculated.

Value

List with the following components:

`Integrated Prediction`	Integrated Prediction based on combining predictions from data subtypes using Least Square Regression
`error_mae`	Mean Absolute Error between Integrated Prediction and Original Output Responses
`error_mse`	Mean Square Error between Integrated Prediction and Original Output Responses
`error_corr`	Pearson Correlation Coefficient between Integrated Prediction and Original Output Responses

See Also

`lsei`

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

IntegratedPrediction *Integrated Prediction of Testing samples from integrated RF or MRF model*

Description

Generates Random Forest or Multivariate Random Forest model for each subtype of dataset and predicts testing samples using the generated models. Subsequently, the prediction for different subtypes of dataset are combined using the Combination weights generated from Integrated Model which is based on Bootstrap error estimate

Usage

```
IntegratedPrediction(finalX, finalY_train, Cell, finalY_train_cell,
  finalY_test_cell, n_tree, m_feature, min_leaf)
```

Arguments

finalX	List of Matrices where each matrix represent a specific data subtype (such as genomic characterizations for drug sensitivity prediction). Each subtype can have different types of features. For example, if there are three subtypes containing 100, 200 and 250 features respectively, finalX will be a list containing 3 matrices of sizes M x 100, M x 200 and M x 250 where M is the number of Samples.
finalY_train	A M x T matrix of output features for training samples, where M is number of samples and T is the number of output features. The dataset is assumed to contain no missing values. If there are missing values, an imputation method should be applied before using the function. A function 'Imputation' is included within the package.
Cell	It contains a list of samples (the samples can be represented either numerically by indices or by names) for each data subtype. For the example of 3 data subtypes, it will be a list containing 3 arrays where each array contains the sample information for each data subtype.
finalY_train_cell	Cell lines of output features for training samples
finalY_test_cell	Cell lines of output features for testing samples
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
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)

Details

Input matrix and output response of training samples have been used to build Random Forest or Multivariate Random Forest model for each subtype of a dataset. These models are used to calculate prediction of testing samples for each subtype separately. Subsequently Combination Weights are used to integrate the predictions from data subtypes.

Combination Weight Generation: For M x N dataset, N number of bootstrap sampling sets are considered. For each bootstrap sampling set and each subtype, a Random Forest (RF) or, Multivariate Random Forest (MRF) model is generated, which is used for calculating the prediction performance for out-of-bag samples. The prediction performance for each dataset subtypes is based on the averaging over different bootstrap training sets. The combination weights (regression coefficients) for each combination of subtypes are generated using least Square Regression from the individual subtype predictions and used to integrate the predictions from data subtypes.

The specific set of combination weights to be used for testing samples will depend on the number of data subtypes available for the testing samples. Note that not all subtype information maybe available for all samples. As an example with three data subtypes, a testing sample with all subtype data available will use the combination weights corresponding to Serial [1 2 3] where as if subtype 3 is not available, the function will use the combination weights corresponding to Serial [1 2].

Value

Final Prediction of testing samples based on provided testing sample names.

Examples

```
library(IntegratedMRF)
data(Dream_Dataset)
Tree=1
Feature=1
Leaf=10
finalX=Dream_Dataset[[1]]
Cell=Dream_Dataset[[2]]
Y_train_Dream=Dream_Dataset[[3]]
Y_train_cell=Dream_Dataset[[4]]
Y_test=Dream_Dataset[[5]]
Y_test_cell=Dream_Dataset[[6]]
Drug=c(1,2,3)
Y_train_Drug=matrix(Y_train_Dream[,Drug],ncol=length(Drug))
IntegratedPrediction(finalX,Y_train_Drug,Cell,Y_train_cell,Y_test_cell,Tree,Feature,Leaf)
```

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(IntegratedMRF)
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 that refers to which child node it will go to using the splitting criteria of the tree node or the 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 splits. 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 1 x N, 1 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

splitt *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

```
splitt(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(IntegratedMRF)
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=splitt(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 (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

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.

Index