Package 'KnowGRRF'

March 6, 2019

Type Package

Title Knowledge-Based Guided Regularized Random Forest

Version 1.0

Date 2019-02-22

Imports randomForest, RRF, PRROC, MASS

Author Original code developed by Xin Guan, modified and added functions developed by Li Liu

Maintainer Xin Guan <xinguan@asu.edu>

License GPL (>= 2)

Description

Random Forest (RF) and Regularized Random Forest can be used for feature selection. Moreover, by Guided Regularized Random Forest, statistical-based weights are used to guide the regularization of random forest and further used for feature selection. This package can integrate prior information from multiple domains (statistical based and knowledge domain) to guide the regularization of random forest and feature selection. For more details, see reference: Guan X., Liu L. (2018) <doi:10.1007/978-3-319-78759-6_1>.

NeedsCompilation no

Repository CRAN

Date/Publication 2019-03-06 11:40:25 UTC

R topics documented:

`get.performance` *Get performance of feature selection*

Description

Comparing feature selected by algorithms and the ground truth in simulation

Usage

```
get.performance(set.truth, set.sel, set.all)
```

Arguments

`set.truth`	A vector of important features set in simulation (ground truth)
`set.sel`	A vector of important features selected by algorithms
`set.all`	A vector of all candidate features for feature selection

Value

returned a vector of feature selection performance including JI (ratio of intersect of two sets and union of two sets), TPR (percentage of correctly selected features in all true important features) and FPR (percentage of wrongly selected features in true non-important features)

Author(s)

Li Liu, Xin Guan

References

Guan, X., & Liu, L. (2018). Know-GRRF: Domain-Knowledge Informed Biomarker Discovery with Random Forests.

Examples

```
set.truth=1:10  ##true important feature from ground truth
set.sel=c(8:10, 95)  ##selected feature by an algorithm
set.all=1:100  ##all candidate features

get.performance(set.truth, set.sel, set.all)
```

on.aic *AIC from model built with KnowGRRF, functions used in optimization to find scaling parameter for rrf.opt.1 or rrf.opt.m*

Description

This function can be used to search scaling parameter used in KnowGRRF while minimizing AIC.

Usage

```
on.aic(X.train, Y.train, pwr, weight, iter=1,total=10, cutoff=0.5, num = 1)
```

Arguments

X.train	a data frame or matrix (like x) containing predictors for the training set.
Y.train	response for the training set. If a factor, classification is assumed, otherwise regression is assumed. If omitted, will run in unsupervised mode.
pwr	Regularization term (a single number for single domain, a vector for multiple domain) to adjust the scale of weights. Larger regularization will differentiate the importance of variables more significantly. Fewer variables tend to be selected with large pwr. This parameter can be found by optimization.
weight	A vector of weights for single domain, or a matrix of weights for multiple domains, corresponding to each of predictors. Weights are between 0 and 1. For multiple domains, each column in weight matrix corresponds to weights from one domain.
iter	The number of RF model built to evaluate AIC. AIC is calculated using out-of-bag prediction from random forest using feature selected.
total	the number of times to repeat the selection for stability test in select.stable function.
cutoff	The minimum percentage of times that the feature is selected in multiple runs for stability test, ranges between 0 and 1.
num	The number of domain knowledge that weights come from.

Value

mean of AIC from a number of RF model using feature selected by KnowGRRF

Author(s)

Xin Guan, Li Liu

References

Guan, X., & Liu, L. (2018). Know-GRRF: Domain-Knowledge Informed Biomarker Discovery with Random Forests.

Examples

```
##used in optim function. See examples in rrf.opt.1 and rrf.opt.m
```

`rf.once` *Build random forest once and return AUC for both training and test set if available*

Description

Build a random forest model once and return AUC for both training prediction (out-of-bag predictions) and test prediction. Work for classification only.

Usage

```
rf.once(X.train, Y.train, X.test, Y.test, fea)
```

Arguments

`X.train`	a data frame or matrix (like x) containing predictors for the training set.
`Y.train`	response for the training set. If a factor, classification is assumed, otherwise regression is assumed. If omitted, will run in unsupervised mode.
`X.test`	an optional data frame or matrix (like x) containing predictors for the test set.
`Y.test`	optional response for the test set.
`fea`	feature index or feature names used to train the model.

Value

return a list, including

`AUC`	AUC calculated from out-of-bag prediction from random forest classification model
`Test.AUC`	AUC calculated from test prediction from random forest classification model. Only available when test set is given

Author(s)

Li Liu, Xin Guan

References

Guan, X., & Liu, L. (2018). Know-GRRF: Domain-Knowledge Informed Biomarker Discovery with Random Forests.

Examples

```
##---- Example: classification ----
library(randomForest)
library(PRROC)
set.seed(1)
X<-data.frame(matrix(rnorm(100*100), nrow=100))
b=seq(0.1, 2.2, 0.2)
##y has a linear relationship with first 10 variables
y=b[6]*X$X5+b[7]*X$X6+b[8]*X$X7+b[9]*X$X8+b[10]*X$X9+b[11]*X$X10
y=as.factor(ifelse(y>0, 1, 0)) ##classification

##split training and test set
X.train=X[1:70,]
X.test=X[71:100,]
y.train=y[1:70]
y.test=y[71:100]

rf.once(X.train, y.train, fea=1:20)  ##no test set
rf.once(X.train, y.train, X.test, y.test, 1:10)  ##relevant feature set
rf.once(X.train, y.train, X.test, y.test, 11:20)  ##irrelevant feature set
```

`rf.repeat` *Build random forest multiple times and return AUC for both training and test set if available*

Description

Due to the randomness of random forest, RF models can be built multiple times to get a better estimation of model performance by assessing AUC on both out-of-bag prediction of training and test predictions on test set. Work for classification only.

Usage

```
rf.repeat(X.train, Y.train, X.test, Y.test, fea, times = 10)
```

Arguments

`X.train`	a data frame or matrix (like x) containing predictors for the training set.
`Y.train`	response for the training set. If a factor, classification is assumed, otherwise regression is assumed. If omitted, will run in unsupervised mode.
`X.test`	a data frame or matrix (like x) containing predictors for the test set.
`Y.test`	response for the test set.
`fea`	feature index or feature names used to train the model.
`times`	the number of RF models built. The more repeats, the less standard error.

Value

return a list, including

AUC	AUC calculated from out-of-bag prediction from random forest classification model
Test.AUC	AUC calculated from test prediction from random forest classification model. Only available when test set is given

Author(s)

Li Liu, Xin Guan

References

Guan, X., & Liu, L. (2018). Know-GRRF: Domain-Knowledge Informed Biomarker Discovery with Random Forests.

Examples

```
##---- Example: classification  ----

set.seed(1)
X<-data.frame(matrix(rnorm(100*100), nrow=100))
b=seq(0.1, 2.2, 0.2)
##y has a linear relationship with first 10 variables
y=b[5]*X$X4+b[6]*X$X5+b[7]*X$X6+b[8]*X$X7+b[9]*X$X8+b[10]*X$X9+b[11]*X$X10
y=as.factor(ifelse(y>0, 1, 0)) ##classification

##split training and test set
X.train=X[1:70,]
X.test=X[71:100,]
y.train=y[1:70]
y.test=y[71:100]

rf.repeat(X.train, y.train, fea=1:20)  ##no test set
rf.repeat(X.train, y.train, X.test, y.test, 1:10)  ##relevant feature set
rf.repeat(X.train, y.train, X.test, y.test, 11:20)  ##irrelevant feature set
```

---

rrf.once *Feature selection by regularized random forest and compare against full model*

Description

Select features using regularized random forest model. Build random forest model either using or not using feature selection. Compare model performance on an independent test set.

Usage

```
rrf.once(X.train, Y.train, X.test, Y.test, coefReg)
```

Arguments

X.train	a data frame or matrix (like x) containing predictors for the training set.
Y.train	response for the training set. If a factor, classification is assumed, otherwise regression is assumed. If omitted, will run in unsupervised mode.
X.test	a data frame or matrix (like x) containing predictors for the test set.
Y.test	response for the test set.
coefReg	regularization coefficient chosen for RRF, ranges between 0 and 1.

Value

return a list, including

perf	number of feature selected by RRF, performance (AUC or MSE depending on classification or regression) of RF model using all features, performance (AUC or MSE depending on classification or regression) of RF model using selected features
FullModel	RF model built with all features
ReducedModel	RF model built with only selected features
featureIndex	feature index selected by RRF

Author(s)

Li Liu, Xin Guan

References

Guan, X., & Liu, L. (2018). Know-GRRF: Domain-Knowledge Informed Biomarker Discovery with Random Forests.

Examples

```
##---- Example: regression ----
library(randomForest)

set.seed(1)
X<-data.frame(matrix(rnorm(100*100), nrow=100))
b=seq(0.1, 2.2, 0.2)
##y has a linear relationship with first 10 variables
y=b[4]*X$X3+b[5]*X$X4+b[6]*X$X5+b[7]*X$X6+b[8]*X$X7+b[9]*X$X8+b[10]*X$X9+b[11]*X$X10


##split training and test set
X.train=X[1:70,]
X.test=X[71:100,]
```

```
y.train=y[1:70]
y.test=y[71:100]

##use RRF to impute regularized coefficients
imp<-randomForest(X.train, y.train)$importance
coefReg=imp/max(imp)

rrf.once(X.train, y.train, X.test, y.test, coefReg)

##---- Example: classification ----
y=as.factor(ifelse(y>0, 1, 0)) ##classification
y.train=y[1:70]
y.test=y[71:100]

##use RRF to impute regularized coefficients
imp<-randomForest(X.train, y.train)$importance
coefReg=imp/max(imp)

rrf.once(X.train, y.train, X.test, y.test, coefReg)
```

`rrf.opt.1` *KnowGRRF with weights from one knowledge domain*

Description

Regularize on the weights to guide RRF feature selection. Weights can from either one knowledge domain, or use statistics-based weights, e.g., p/q value, variable importance, etc. Feature set selected is also based on stability, that is the frequency of selection from multiple runs. Features that are consistently selected from multiple runs will be used in a random forest model, from which AIC and AUC will be calculated to evaluate the model performance. Only AIC will be calculated for regression.

Usage

```
rrf.opt.1(X.train, Y.train, X.test=NULL, Y.test=NULL, pwr, weight,
iter=1,total=10, cutoff=0.5)
```

Arguments

`X.train`	a data frame or matrix (like x) containing predictors for the training set.
`Y.train`	response for the training set. If a factor, classification is assumed, otherwise regression is assumed. If omitted, will run in unsupervised mode.
`X.test`	an optional data frame or matrix (like x) containing predictors for the test set.
`Y.test`	optional response for the test set.

pwr	Regularization term to adjust the scale of weights. When one domain knowledge is used, this is a single regularization value. Larger regularization will differentiate the importance of variables more significantly. Fewer variables tend to be selected with large pwr. This parameter can be tuned using optimization methods or grid searching with on.aic function.
weight	A vector of weights corresponding to each of predictors. Weights are between 0 and 1.
iter	The number of RF model built to evaluate AIC and AUC. AIC is calculated using out-of-bag prediction from random forest using feature selected. AUC is calculated for classification problem only.
total	the number of times to repeat the selection for stability test in select.stable function.
cutoff	The minimum percentage of times that the feature is selected in multiple runs for stability test, ranges between 0 and 1.

Value

return a list, including

AIC	AIC calculated from random forest model out-of-bag predicted probability for classification, or out-of-bag prediction for classification
AUC	AUC calculated from out-of-bag prediction from random forest classification model
Test.AUC	AUC calculated from test prediction from random forest classification model
AUC	AUC calculated from out-of-bag prediction from random forest classification model
feaSet	feature set selected

Note

This function can be used after weights and regularization term are determined. Weights are from knowledege domain and regularization term can be determined by optimization. See example.

Author(s)

Li Liu, Xin Guan

References

Guan, X., & Liu, L. (2018). Know-GRRF: Domain-Knowledge Informed Biomarker Discovery with Random Forests.

Examples

```
##---- Example: classification ----
library(randomForest)

set.seed(1)
```

```
X.train<-data.frame(matrix(rnorm(10*100), nrow=100))
b=seq(1, 6, 0.5)
##y has a linear relationship with 5 variables
y.train=b[7]*X.train$X6+b[8]*X.train$X7+b[9]*X.train$X8+b[10]*X.train$X9+b[11]*X.train$X10
y.train=as.factor(ifelse(y.train>0, 1, 0)) ##classification

##use weights from domain knowledge. If not available,
##can use statistic-based weights, e.g., variable importance, p/q value, etc
imp<-randomForest(X.train, y.train)$importance
coefReg=0.5+0.5*imp/max(imp)


#'\donttest{
#'use optimization function to find the appropriate regularization term
#'to scale weights and then apply the weights to guide the RRF

#'opt<-optim(par=5, fn=on.aic, X.train=X.train, Y.train=y.train,
#'weight=coefReg, iter=5,total=10, cutoff=0.5, num = 1, method='L-BFGS-B',
#'lower=1, upper=10, control=list(fnscale=1,factr=100, trace = TRUE))
#'gives an error because under the initial value and searching space,
#'no feature is selected, could try smaller number for initialization


#'opt<-optim(par=1, fn=on.aic, X.train=X.train, Y.train=y.train,
#'weight=coefReg, iter=5,total=10, cutoff=0.5, num = 1, method='L-BFGS-B',
#'lower=0.01, upper=0.5, control=list(fnscale=1,factr=100, trace = TRUE))
#'converged, could take long to run, opt$par returns pwr that could be used in rrf.opt.1
#'}

rrf.opt.1(X.train, y.train, pwr=1, weight=coefReg, total=5)
```

`rrf.opt.m` *KnowGRRF with weights from multiple knowledge domain*

Description

Regularize on the weights to guide RRF feature selection. Weights can from multiple knowledge domain and/or combination with statistics-based weights, e.g., p/q value, variable importance, etc. Proportion of weights can be scaled by regularization parameters. Feature set selected is also based on stability, that is the frequency of selection from multiple runs. Features that are consistently selected from multiple runs will be used in a random forest model, from which AIC and AUC will be calculated to evaluate the model performance. Only AIC will be calculated for regression.

Usage

```
rrf.opt.m(X.train, Y.train, X.test=NULL, Y.test=NULL, pwr,
weight, iter=1,total=10, cutoff=0.5)
```

Arguments

X.train	a data frame or matrix (like x) containing predictors for the training set.
Y.train	response for the training set. If a factor, classification is assumed, otherwise regression is assumed. If omitted, will run in unsupervised mode.
X.test	an optional data frame or matrix (like x) containing predictors for the test set.
Y.test	optional response for the test set.
pwr	Regularization term to adjust the scale of weights. When multiple domain knowledge is used, pwr is a vector, with length equal to the number of domain knowledge plus one. First parameter is the scaling parameter, and the rest of the vectors correspond to the relative importance of each domain knowledge. Larger regularization will differentiate the importance of variables more significantly. Fewer variables tend to be selected with large pwr. This parameter can be tuned using optimization methods or grid searching with on.aic function.
weight	A matrix of weights corresponding to each of predictors. Each column correspond to each domain knowledge and each row correspond to each variable. Weights are between 0 and 1.
iter	The number of RF model built to evaluate AIC and AUC. AIC is calculated using out-of-bag prediction from random forest using feature selected. AUC is calculated for classification problem only.
total	the number of times to repeat the selection for stability test in select.stable function.
cutoff	The minimum percentage of times that the feature is selected in multiple runs for stability test, ranges between 0 and 1.

Value

return a list, including

AIC	AIC calculated from random forest model out-of-bag predicted probability for classification, or out-of-bag prediction for classification
AUC	AUC calculated from out-of-bag prediction from random forest classification model
Test.AUC	AUC calculated from test prediction from random forest classification model
AUC	AUC calculated from out-of-bag prediction from random forest classification model
feaSet	feature set selected

Note

This function can be used after weights and regularization term are determined. Weights are from knowledege domain and regularization term can be determined by optimization. See example.

Author(s)

Xin Guan, Li Liu

References

Guan, X., & Liu, L. (2018). Know-GRRF: Domain-Knowledge Informed Biomarker Discovery with Random Forests.

Examples

```
##---- Example: regression ----
library(randomForest)

set.seed(1)
X.train<-data.frame(matrix(rnorm(100*100), nrow=100))
b=seq(1, 6, 0.5)
##y has a linear relationship with first 10 variables
y.train=b[7]*X.train$X6+b[8]*X.train$X7+b[9]*X.train$X8+b[10]*X.train$X9+b[11]*X.train$X10


##use weights from domain knowledge. If not available,
##can use statistic-based weights, e.g., variable importance, p/q value, etc
prior1 <- abs(c(rnorm(5, 5, 1), rnorm(95, 0, 1)))
##domain 1 suggest first five are important variables
prior2 <- abs(c(rnorm(5, 0, 1), rnorm(5, 8, 2), rnorm(90, 0, 1)))
##domain 2 suggest next five are important variables
imp<-randomForest(X.train, y.train)$importance
prior3=0.5+0.5*imp/max(imp)   ##domain 3 uses relative varialbe importance



#'\donttest{
#'use optimization function to find the appropriate regularization term
#'to scale weights and then apply the weights to guide the RRF

#'opt<-optim(par=c(1,1,1,1), fn=on.aic, X.train=X.train, Y.train=y.train,
#'weight=cbind(prior1, prior2, prior3), iter=5,total=10, cutoff=0.5, num = 3,
#'method='L-BFGS-B', lower=0.01, upper=0.5, control=list(fnscale=1,trace = TRUE))
#'can take long for four parameters to be optimized.
#'opt$par can be used as input of pwr in rrf.opt.m
#'}

rrf.opt.m(X.train, y.train, pwr=c(5,1,1,1), weight=cbind(prior1, prior2, prior3))
```

`select.stable` *Select a set of stable features based on frequency picked by GRRF.*

Description

Perform feature selection by GRRF. Repeat it multiple times to select a stable set of features that are consistently selected according to the selection frequency.

Usage

```
select.stable(X.train, Y.train, coefReg, total=10, cutoff=0.5)
```

Arguments

X.train	a data frame or matrix (like x) containing predictors for the training set.
Y.train	response for the training set. If a factor, classification is assumed, otherwise regression is assumed. If omitted, will run in unsupervised mode.
coefReg	regularization coefficient chosen for RRF, ranges between 0 and 1.
total	the number of times to repeat the process.
cutoff	The minimum percentage of times that the feature is selected by RRF, ranges between 0 and 1.

Value

a stable set of features selected by GRRF

Note

For customized hyperparameter setting, can directly call RRF function from RRF package repeatly in a for loop.

Author(s)

Li Liu, Xin Guan

References

Guan, X., & Liu, L. (2018). Know-GRRF: Domain-Knowledge Informed Biomarker Discovery with Random Forests.

Examples

```
##---- Example: classification ----
library(randomForest)

set.seed(1)
X.train<-data.frame(matrix(rnorm(100*100), nrow=100))
b=seq(0.1, 2.2, 0.2)
##y has a linear relationship with first 10 variables
y.train=b[7]*X.train$X6+b[8]*X.train$X7+b[9]*X.train$X8+b[10]*X.train$X9+b[11]*X.train$X10
y.train=ifelse(y.train>0, 1, 0) ##classification

##use RRF to impute regularized coefficients
imp<-randomForest(X.train, as.factor(y.train))$importance
coefReg=0.5+0.5*imp/max(imp)

##select a stable set of feature that are consistently selected more than half of times
select.stable(X.train, as.factor(y.train), coefReg)
```

`select.stable.aic` *Select a set of stable features based on AIC after an initial selection by GRRF*

Description

Perform feature selection by GRRF and followed by stepwise model selection by AIC. Repeat it multiple times to select a stable set of features that are selected according to AIC.

Usage

```
select.stable.aic(X.train, Y.train, coefReg, total=10)
```

Arguments

`X.train`	a data frame or matrix (like x) containing predictors for the training set.
`Y.train`	response for the training set. If a factor, classification is assumed, otherwise regression is assumed. If omitted, will run in unsupervised mode.
`coefReg`	regularization coefficient chosen for RRF, ranges between 0 and 1.
`total`	the number of times to repeat the process.

Value

a stable set of features selected by GRRF

Note

For customized hyperparameter setting, can directly call RRF function from RRF package repeatly in a for loop.

Author(s)

Li Liu, Xin Guan

References

Guan, X., & Liu, L. (2018). Know-GRRF: Domain-Knowledge Informed Biomarker Discovery with Random Forests.

Examples

```
##---- Example: classification  ----
library(randomForest)

set.seed(1)
X.train<-data.frame(matrix(rnorm(100*100), nrow=100))
b=seq(0.1, 2.2, 0.2)
##y has a linear relationship with first 10 variables
y.train=b[7]*X.train$X6+b[8]*X.train$X7+b[9]*X.train$X8+b[10]*X.train$X9+b[11]*X.train$X10
y.train=ifelse(y.train>0, 1, 0) ##classification

##use RRF to impute regularized coefficients
imp<-randomForest(X.train, as.factor(y.train))$importance
coefReg=0.5+0.5*imp/max(imp)

##select a stable set of feature that are selected by GRRF followed by stepAIC
select.stable.aic(X.train, as.factor(y.train), coefReg)
```

`write.roc` *write test ROC to a data table.*

Description

write a data table including False Positive Rate, True Positive Rate and cutoff on test dataset. Work for classification only.

Usage

```
write.roc(X.train, Y.train, X.test, Y.test, fea, file.name="")
```

Arguments

`X.train`	a data frame or matrix (like x) containing predictors for the training set.
`Y.train`	response for the training set. If a factor, classification is assumed, otherwise regression is assumed. If omitted, will run in unsupervised mode.
`X.test`	a data frame or matrix (like x) containing predictors for the test set.
`Y.test`	response for the test set.
`fea`	feature index or feature names used to train the model.
`file.name`	directory and name of files that write to. If directory is not given, will write to working directory.

Value

a data table in csv format which columns FPR, TPR and cutoff.

Author(s)

Li Liu, Xin Guan

References

Guan, X., & Liu, L. (2018). Know-GRRF: Domain-Knowledge Informed Biomarker Discovery with Random Forests.

Examples

```
##---- Example: classification ----

set.seed(1)
X<-data.frame(matrix(rnorm(100*100), nrow=100))
b=seq(0.1, 2.2, 0.2)
##y has a linear relationship with first 10 variables
y=b[7]*X$X6+b[8]*X$X7+b[9]*X$X8+b[10]*X$X9+b[11]*X$X10
y=as.factor(ifelse(y>0, 1, 0)) ##classification

##split training and test set
X.train=X[1:70,]
X.test=X[71:100,]
y.train=y[1:70]
y.test=y[71:100]

##save to a temp file
write.roc(X.train, y.train, X.test, y.test, fea=1:20, paste(tempdir(), "example.csv", sep="/"))
```

Index