Package 'SparseLearner'

November 17, 2015

Title Sparse Learning Algorithms Using a LASSO-Type Penalty for Coefficient Estimation and Model Prediction

Version 1.0-2

Author Pi Guo, Yuantao Hao

Maintainer Pi Guo <guopi.01@163.com>

Description Coefficient estimation and model prediction based on the LASSO sparse learning algorithm and its improved versions such as Bolasso, bootstrap ranking LASSO, two-stage hybrid LASSO and others. These LASSO estimation procedures are applied in the fields of variable selection, graphical modeling and ensemble learning. The bagging LASSO model uses a Monte Carlo cross-entropy algorithm to determine the best base-level models and improve predictive performance.

Depends R (>= 3.0.2), glmnet

Imports SIS, mlbench, RankAggreg, SiZer, lqa, qgraph

License GPL-2

LazyData true

URL https://www.researchgate.net/profile/Pi_Guo3

NeedsCompilation no

Repository CRAN

Date/Publication 2015-11-17 14:40:37

R topics documented:

`Bagging.lasso` *A Bagging Prediction Model Using LASSO Selection Algorithm.*

Description

This function performs a bagging prediction for linear and logistic regression model using the LASSO selection algorithm.

Usage

```
Bagging.lasso(x, y, family = c("gaussian", "binomial"), M = 100, subspace.size = 10,
          predictor.subset = round((9/10) * ncol(x)), boot.scale = 1, kfold = 10,
             predictor.importance = TRUE, trimmed = FALSE, weighted = TRUE,
             verbose = TRUE, seed = 123)
```

Arguments

`x`	input matrix. The dimension of the matrix is nobs x nvars; each row is a vector of observations of the variables.
`y`	response variable. For family="gaussian", y is a vector of quantitative response. For family="binomial" should be a factor with two levels '0' and '1' and the level of '1' is the target class.
`family`	response type (see above).
`M`	the number of base-level models (LASSO linear or logistic regression models) to obtain a final prediction. Note that it also corresponds to the number of bootstrap samples to draw. Defaults to 100.
`subspace.size`	the number of random subspaces to construct an ensemble prediction model. Defaults to 10.
`predictor.subset`	the subset of randomly selected predictors from the training set to reduce the original p-dimensional feature space. Defaults to (9/10)*ncol(x) where ncol(x) represents the the original p-dimensional feature space of input matrix x.
`boot.scale`	the scale of sample size in each bootstrap re-sampling, relative to the original sample size. Defaults to 1.0, equaling to the original size of training samples.
`kfold`	the number of folds of cross validation - default is 10. Although kfold can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is kfold=3.
`predictor.importance`	logical. Should the importance of each predictor in the bagging LASSO model be evaluated? Defaults to TRUE. A permutation-based variable importance measure estimated by the out-of-bag error rate is adapted for the bagging model.
`trimmed`	logical. Should a trimmed bagging strategy be performed? Defaults to FALSE. Traditional bagging draws bootstrap samples from the training sample, applies

	the base-level model to each bootstrap sample, and then averages over all obtained prediction rules. The idea of trimmed bagging is to exclude the bootstrapped prediction rules that yield the highest error rates and to aggregate over the remaining ones.
`weighted`	logical. Should a weighted rank aggregation procedure be performed? Defaults to TRUE. This procedure uses a Monte Carlo cross-entropy algorithm combining the ranks of a set of based-level model under consideration via a weighted aggregation that optimizes a distance criterion to determine the best performance base-level model.
`verbose`	logical. Should the iterative process information of bagging model be presented? Defaults to TRUE.
`seed`	the seed for random sampling, with the default value 0123.

Details

This bagging LASSO model Bagging.lasso generates an ensemble prediction based on the L1-regularized linear or logistic regression models. The Bagging.lasso function uses a Monte Carlo cross-entropy algorithm to combine the ranks of a set of based-level LASSO regression model under consideration via a weighted aggregation to determine the best base-level model. In the Bagging.lasso, the glmnet algorithm is performed to fit LASSO model paths for linear and logistic regression using coordinate descent. A random subspace method is employed to improve the predictive performance. In addition, a strategy of trimmed bagging can be defined to exclude the bootstrapped prediction rules that yield the highest error rates and to aggregate over the remaining prediction rules.

Value

`family`	the response type.
`M`	the number of base-level models to obtain a bagging prediction.
`predictor.subset`	the subset of randomly selected predictors from the training set to reduce the original p-dimensional feature space.
`subspace.size`	the number of random subspaces to construct an ensemble prediction model.
`validation.metric`	the model validation measures.
`boot.scale`	the scale of sample size in each bootstrap re-sampling, relative to the original sample size.
`distance`	the distance function used in the weighted aggregation to define the similarity between each two sets of based-level model.
`models.fitted`	the base-level LASSO regression models fitted by the Bagging.lasso function.
`models.trimmed`	the trimmed base-level models fitted by the Bagging.lasso function if the trimmed bagging strategy is performed.
`y.true`	the true values of reponse vector y.
`conv.scores`	the score matrix generated in the Monte Carlo cross-entropy algorithm according to the validation measures defined.
`importance`	the importance socres of variables identified by the Bagging.lasso model.

References

[1] Guo, P., Zeng, F., Hu, X., Zhang, D., Zhu, S., Deng, Y., & Hao, Y. (2015). Improved Variable Selection Algorithm Using a LASSO-Type Penalty, with an Application to Assessing Hepatitis B Infection Relevant Factors in Community Residents. PLoS One, 27;10(7):e0134151.

[2] Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. Journal of the royal statistical society series B (statistical methodology), 73(3):273-282.

[3] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.

Examples

```
# Example 1: Bagging LASSO linear regression model.
library(mlbench)
set.seed(0123)
mydata <- mlbench.threenorm(100, d=10)
x <- mydata$x
y <- mydata$classes
mydata <- as.data.frame(cbind(x, y))
colnames(mydata) <- c(paste("A", 1:10, sep=""), "y")
mydata$y <- ifelse(mydata$y==1, 0, 1)
# Split into training and testing data.
S1 <- as.vector(which(mydata$y==0))
S2 <- as.vector(which(mydata$y==1))
S3 <- sample(S1, ceiling(length(S1)*0.8), replace=FALSE)
S4 <- sample(S2, ceiling(length(S2)*0.8), replace=FALSE)
TrainInd <- c(S3, S4)
TestInd <- setdiff(1:length(mydata$y), TrainInd)
TrainXY <- mydata[TrainInd, ]
TestXY <- mydata[TestInd, ]
# Fit a bagging LASSO linear regression model, where the parameters
# of M in the following example is set as small values to reduce the
# running time, however the default value is proposed.
Bagging.fit <- Bagging.lasso(x=TrainXY[, -10], y=TrainXY[, 10],
family=c("gaussian"), M=2, predictor.subset=round((9/10)*ncol(x)),
predictor.importance=TRUE, trimmed=FALSE, weighted=TRUE, seed=0123)
# Print a 'bagging' object fitted by the Bagging.fit function.
Print.bagging(Bagging.fit)
# Make predictions from a bagging LASSO linear regression model.
pred <- Predict.bagging(Bagging.fit, newx=TestXY[, -10], y=NULL, trimmed=FALSE)
pred
# Generate the plot of variable importance.
Plot.importance(Bagging.fit)
# Example 2: Bagging LASSO logistic regression model.
library(mlbench)
set.seed(0123)
mydata <- mlbench.threenorm(100, d=10)
x <- mydata$x
y <- mydata$classes
mydata <- as.data.frame(cbind(x, y))
colnames(mydata) <- c(paste("A", 1:10, sep=""), "y")
mydata$y <- ifelse(mydata$y==1, 0, 1)
# Split into training and testing data.
```

```
S1 <- as.vector(which(mydata$y==0))
S2 <- as.vector(which(mydata$y==1))
S3 <- sample(S1, ceiling(length(S1)*0.8), replace=FALSE)
S4 <- sample(S2, ceiling(length(S2)*0.8), replace=FALSE)
TrainInd <- c(S3, S4)
TestInd <- setdiff(1:length(mydata$y), TrainInd)
TrainXY <- mydata[TrainInd, ]
TestXY <- mydata[TestInd, ]
# Fit a bagging LASSO logistic regression model, where the parameters
# of M in the following example is set as small values to reduce the
# running time, however the default value is proposed.
Bagging.fit <- Bagging.lasso(x=TrainXY[, -11], y=TrainXY[, 11],
family=c("binomial"), M=2, predictor.subset=round((9/10)*ncol(x)),
predictor.importance=TRUE, trimmed=FALSE, weighted=TRUE, seed=0123)
# Print a 'bagging' object fitted by the Bagging.fit function.
Print.bagging(Bagging.fit)
# Make predictions from a bagging LASSO logistic regression model.
pred <- Predict.bagging(Bagging.fit, newx=TestXY[, -11], y=NULL, trimmed=FALSE)
pred
# Generate the plot of variable importance.
Plot.importance(Bagging.fit)
```

`Bolasso` *Bolasso model.*

Description

This function performs a Bolasso logistic regression model and produces an optimal set of predictors.

Usage

```
Bolasso(x, y, BM = 100, kfold = 10, seed = 0123)
```

Arguments

`x`	predictor matrix.
`y`	response variable, a factor object with values of 0 and 1.
`BM`	the number of bootstrapping, with the default value 100.
`kfold`	the number of folds of cross validation - default is 10. Although kfold can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is kfold=3.
`seed`	the seed for random sampling, with the default value 0123.

Details

This function runs the LASSO logistic regression model using several bootstrap samples of the original data, and then intersects the non-zero coefficients for estimating consistent coefficients. A specific value of BM parameter should be supplied, however BM=100 is proposed by default. Users can reduce the running time by using 3-fold CV, but the proposed 10-fold CV is assumed by default.

Value

BM	the number of bootstrapping in this procedure.
var.selected	significant variables that are selected by the Bolasso model.

References

[1] Friedman, J., Hastie, T. and Tibshirani, R. (2008). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22.

[2] Bach, F.R. (2008). Bolasso: model consistent lasso estimation through the bootstrap. Proceedings of the 25th international conference on Machine learning. ACM. pp. 33:40.

Examples

```
library(datasets)
head(iris)
X <- as.matrix(subset(iris, iris$Species!="setosa")[, -5])
Y <- as.factor(ifelse(subset(iris, iris$Species!="setosa")[, 5]=='versicolor', 0, 1))
# Fit a Bolasso logistic regression model
# The BM parameter in the following example is set as small value to reduce
# the running time, however the default value is proposed
Bolasso.fit <- Bolasso(x=X, y=Y, BM=5, seed=0123)
# Significant variables that are selected by the Bolasso model
Bolasso.fit$var.selected
```

BRLasso *Bootstrap ranking LASSO model.*

Description

This function performs a LASSO logistic regression model using a bootstrap ranking procedure, namely the BRLasso logistic regression model, produces an optimal set of predictors and returns the robust estimations of coefficients of the selected predictors.

Usage

```
BRLasso(x, y, B = 5, Boots = 100, kfold = 10, seed = 0123)
```

Arguments

x	predictor matrix.
y	response variable, a factor object with values of 0 and 1.
B	the number of external loop for intersection operation, with the default value 5.
Boots	the number of internal loop for bootstrap sampling, with the default value 100.
kfold	the number of folds of cross validation - default is 10. Although kfold can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is kfold=3.
seed	the seed for random sampling, with the default value 0123.

Details

This function runs the LASSO logistic regression model using a bootstrap ranking procedure. The bootstrap ranking procedure generates a LASSO estimates matrix representing variable ranking according to importance, and runs the external intersection operation to extract a panel of informative variables. Like the Bolasso, the bootstrap ranking procedure performs the intersection operation to extract relevant variables from sufficiently many different data sets using the sample bootstrapping method. However, instead of directly intersecting several sets of non-zero LASSO estimates, the model generates a LASSO estimate matrix representing variable ranking according to their importance, and then intersects to obtain a robust result. During each internal loop, bootstrap samples are generated by randomly selecting n individuals with replacement from a given data set of n individuals. For each such sample, LASSO regression coefficients are estimated for all variables, and the average estimation across internal bootstraps is calculated as the measurement of variable importance. Specific values of B and Boots parameters should be supplied, however B=5 and Boots=100 is proposed by default. Users can reduce the running time by using 3-fold CV, but the proposed 10-fold CV is assumed by default.

Value

`B`	the number of external loop for intersection operation.
`Boots`	the number of internal loop for bootstrap sampling.
`var.selected`	significant variables that are selected by the BRLasso model.
`var.coef`	coefficients of the selected significant variables.

References

[1] Guo, P., Zeng, F., Hu, X., Zhang, D., Zhu, S., Deng, Y., & Hao, Y. (2015). Improved Variable Selection Algorithm Using a LASSO-Type Penalty, with an Application to Assessing Hepatitis B Infection Relevant Factors in Community Residents. PLoS One, 27;10(7):e0134151.

[2] Friedman, J., Hastie, T. and Tibshirani, R. (2008). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22.

Examples

```
library(datasets)
head(iris)
X <- as.matrix(subset(iris, iris$Species!="setosa")[, -5])
Y <- as.factor(ifelse(subset(iris, iris$Species!="setosa")[, 5]=='versicolor', 0, 1))
# Fit a bootstrap ranking LASSO (BRLasso) logistic regression model.
# The parameters of B and Boots in the following example are set as small values to
# reduce the running time, however the default values are proposed.
BRLasso.fit <- BRLasso(x=X, y=Y, B=2, Boots=5, seed=0123)
# Significant variables that are selected by the BRLasso model.
BRLasso.fit$var.selected
# Coefficients of the selected variables.
BRLasso.fit$var.coef
```

`Plot.importance` *Generate a plot of variable importance.*

Description

This function generates a plot for evaluating variable importance based on a bagging object fitted by the bagging.lasso model.

Usage

```
Plot.importance(x, max.var.show = 40, xlab = "Importance Score", ylab = NULL,
                main = "Variable Importance Plot")
```

Arguments

`x`	a fitted bagging object.
`max.var.show`	the maximum number of variables to be shown in the plot. Defaults to 40.
`xlab`	a title for the x axis.
`ylab`	a title for the y axis.
`main`	an overall title for the plot.

Details

A plot of variable importance based on the Bagging.lasso model is produced, and nothing is returned.

References

[1] Guo, P., Zeng, F., Hu, X., Zhang, D., Zhu, S., Deng, Y., & Hao, Y. (2015). Improved Variable Selection Algorithm Using a LASSO-Type Penalty, with an Application to Assessing Hepatitis B Infection Relevant Factors in Community Residents. PLoS One, 27;10(7):e0134151.

[2] Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. Journal of the royal statistical society series B (statistical methodology), 73(3):273-282.

[3] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.

Examples

```
library(mlbench)
set.seed(0123)
mydata <- mlbench.threenorm(100, d=10)
x <- mydata$x
y <- mydata$classes
mydata <- as.data.frame(cbind(x, y))
colnames(mydata) <- c(paste("A", 1:10, sep=""), "y")
mydata$y <- ifelse(mydata$y==1, 0, 1)
# Split into training and testing data.
```

```
S1 <- as.vector(which(mydata$y==0))
S2 <- as.vector(which(mydata$y==1))
S3 <- sample(S1, ceiling(length(S1)*0.8), replace=FALSE)
S4 <- sample(S2, ceiling(length(S2)*0.8), replace=FALSE)
TrainInd <- c(S3, S4)
TestInd <- setdiff(1:length(mydata$y), TrainInd)
TrainXY <- mydata[TrainInd, ]
TestXY <- mydata[TestInd, ]
# Fit a bagging LASSO linear regression model, where the parameters
# of M in the following example is set as small values to reduce the
# running time, however the default value is proposed.
Bagging.fit <- Bagging.lasso(x=TrainXY[, -10], y=TrainXY[, 10],
family=c("gaussian"), M=2, predictor.subset=round((9/10)*ncol(x)),
predictor.importance=TRUE, trimmed=FALSE, weighted=TRUE, seed=0123)
Plot.importance(Bagging.fit)
```

`Predict.bagging` *Make predictions for new data from a 'bagging' object.*

Description

This function makes predictions for new data from a bagging LASSO linear or logistic regression model, using the stored 'bagging' object, with or without the use of trimmed bagging strategy.

Usage

```
Predict.bagging(object, newx, y = NULL, trimmed = FALSE, scale.trimmed = 0.75)
```

Arguments

`object`	a fitted 'bagging' object.
`newx`	matrix of new values for x at which predictions are to be made. Must be a matrix. See documentation for Bagging.lasso.
`y`	response variable. Defaults to NULL. If the response variable for the newx matrix is known and input, the corresponding validation measures can be calculated for evaluating prediction performance.
`trimmed`	logical. Should a trimmed bagging strategy be performed? Defaults to FALSE. This argument should correspond to the same setting in the Bagging.lasso function. See documentation for Bagging.lasso.
`scale.trimmed`	the portion to trim of the "worst" based-level models, in the sense of having the largest error rates, and to average only over the most accurate base-level models. Defaults to 0.75.

Details

This function makes a prediction based on the object fitted by the Bagging.lasso model.

Value

`y.new` the predicted values of response vector y.

`probabilities` the predicted probabilities of response vector y.

`predicted.matrix`
the matrix of predicted values of response vector y based on the base-level LASSO regression models.

`bagging.prediction`
the performance of bagging prediction accordig to the model validation measures defined.

References

[1] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.

[2] Croux, C., Joossens, K., & Lemmens, A. (2007). Trimmed bagging. Computational Statistics & Data Analysis, 52(1), 362-368.

Examples

```
library(mlbench)
set.seed(0123)
mydata <- mlbench.threenorm(100, d=10)
x <- mydata$x
y <- mydata$classes
mydata <- as.data.frame(cbind(x, y))
colnames(mydata) <- c(paste("A", 1:10, sep=""), "y")
mydata$y <- ifelse(mydata$y==1, 0, 1)
# Split into training and testing data.
S1 <- as.vector(which(mydata$y==0))
S2 <- as.vector(which(mydata$y==1))
S3 <- sample(S1, ceiling(length(S1)*0.8), replace=FALSE)
S4 <- sample(S2, ceiling(length(S2)*0.8), replace=FALSE)
TrainInd <- c(S3, S4)
TestInd <- setdiff(1:length(mydata$y), TrainInd)
TrainXY <- mydata[TrainInd, ]
TestXY <- mydata[TestInd, ]
# Fit a bagging LASSO linear regression model, where the parameters
# of M in the following example is set as small values to reduce the
# running time, however the default value is proposed.
Bagging.fit <- Bagging.lasso(x=TrainXY[, -10], y=TrainXY[, 10],
family=c("gaussian"), M=2, predictor.subset=round((9/10)*ncol(x)),
predictor.importance=TRUE, trimmed=FALSE, weighted=TRUE, seed=0123)
Bagging.fit
# Make predictions from a bagging LASSO linear regression model.
pred <- Predict.bagging(Bagging.fit, newx=TestXY[, -10], y=NULL)
pred
```

Print.bagging *Print a bagging object.*

Description

This function prints a summary of the bagging object fitted by the bagging.lasso function.

Usage

```
Print.bagging(x)
```

Arguments

x	a fitted bagging object.

Details

The call that produced the object Bagging.lasso is printed.

References

[1] Guo, P., Zeng, F., Hu, X., Zhang, D., Zhu, S., Deng, Y., & Hao, Y. (2015). Improved Variable Selection Algorithm Using a LASSO-Type Penalty, with an Application to Assessing Hepatitis B Infection Relevant Factors in Community Residents. PLoS One, 27;10(7):e0134151.

[2] Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. Journal of the royal statistical society series B (statistical methodology), 73(3):273-282.

[3] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.

Examples

```
library(mlbench)
set.seed(0123)
mydata <- mlbench.threenorm(100, d=10)
x <- mydata$x
y <- mydata$classes
mydata <- as.data.frame(cbind(x, y))
colnames(mydata) <- c(paste("A", 1:10, sep=""), "y")
mydata$y <- ifelse(mydata$y==1, 0, 1)
# Split into training and testing data.
S1 <- as.vector(which(mydata$y==0))
S2 <- as.vector(which(mydata$y==1))
S3 <- sample(S1, ceiling(length(S1)*0.8), replace=FALSE)
S4 <- sample(S2, ceiling(length(S2)*0.8), replace=FALSE)
TrainInd <- c(S3, S4)
TestInd <- setdiff(1:length(mydata$y), TrainInd)
TrainXY <- mydata[TrainInd, ]
TestXY <- mydata[TestInd, ]
# Fit a bagging LASSO linear regression model, where the parameters
```

```
# of M in the following example is set as small values to reduce the
# running time, however the default value is proposed.
Bagging.fit <- Bagging.lasso(x=TrainXY[, -10], y=TrainXY[, 10],
family=c("gaussian"), M=2, predictor.subset=round((9/10)*ncol(x)),
predictor.importance=FALSE, trimmed=FALSE, weighted=TRUE, seed=0123)
# Print a 'bagging' object fitted by the Bagging.fit function.
Print.bagging(Bagging.fit)
```

`Sparse.graph` *Graphic Modeling Using LASSO-Type Sparse Learning Algorithm.*

Description

This function builds a gaussian or binary graph based on the bootstrap ranking LASSO regression method.

Usage

```
Sparse.graph(x, graph.type = c("gaussian"), B = 5, Boots = 100, edge.rule = c("AND"),
             kfold = 10, plot = TRUE, seed = 0123)
```

Arguments

`x`	input matrix. The dimension of the matrix is nobs x nvars; each row is a vector of observations of the variables. Gaussian or binary data is supported.
`graph.type`	the type of gaussian or binary graph. Defaults to gaussian.
`B`	the number of external loop for intersection operation. Defaults to 5.
`Boots`	the number of internal loop for bootstrap sampling. Defaults to 100.
`edge.rule`	the rule indicating whether the AND-rule or the OR-rule should be used to define the edges in the graph. Defaults to AND.
`kfold`	the number of folds of cross validation - default is 10. Although kfold can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is kfold=3.
`plot`	logical. Should the resulting graph be plotted? Defaults to TRUE.
`seed`	the seed for random sampling, with the default value 0123.

Details

This graph estimation procedure Sparse.graph, which is based on the L1-regularized regression model, combines a bootstrap ranking strategy with model selection using the glmnet algorithm. The glmnet algorithm fits LASSO model paths for linear and logistic regression using coordinate descent. Thus, the Sparse.graph procedure identifies relevant relationships between gaussian and binary variables, and assesses network structures from data. The resulting graph consists of variables as nodes and relevant relationships as edges. The combination of the LASSO penalized regression model and a bootstrap ranking strategy demonstrates a higher power and a lower false positive rate during variable selection, and is proposed to identify significant association between variables in epidemiological analysis.

Value

adj.matrix	the adjacency matrix.
graph.type	the type of graph. Currently, this procedure is supported for gaussian and binary data.
B	the number of external loop for intersection operation.
Boots	the number of internal loop for bootstrap sampling.
edge.rule	the rule used to define the edges in the graph.

References

[1] Guo, P., Zeng, F., Hu, X., Zhang, D., Zhu, S., Deng, Y., & Hao, Y. (2015). Improved Variable Selection Algorithm Using a LASSO-Type Penalty, with an Application to Assessing Hepatitis B Infection Relevant Factors in Community Residents. PLoS One, 27;10(7):e0134151.

[2] Friedman, J., Hastie, T. and Tibshirani, R. (2008). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22.

[3] Strobl, R., Grill, E., Mansmann, U. (2012). Graphical modeling of binary data using the LASSO: a simulation study. BMC Medical Research Methodology, 12:16.

[4] Meinshausen, N., Buehlmann, P. (2006). High-dimensional graphs and variable selection with the Lasso. Ann Stat, 34:1436-1462.

Examples

```
# Example 1: Gene network estimation using the bootstrap ranking LASSO method.
# Gaussian graph with OR-rule.
library(SIS)
data(leukemia.train)
# Genes screened by the LASSO algorithm as candidates for graphical modeling.
x <- as.matrix(leukemia.train[, -7130])
y <- as.numeric(leukemia.train[, 7130])
set.seed(0123)
cvfit <- cv.glmnet(x=x, y=y, type.measure="deviance", nfolds=3, family="binomial")
model.final <- cvfit$glmnet.fit
nzero <- as.matrix(coef(model.final, s=cvfit$lambda.min))
# To reduce the running time, only half of significant genes are shown.
var_nz <- sort(abs(nzero[nzero[,1]!=0, ][-1]), decreasing=TRUE)
var_nz <- names(var_nz[1:(length(var_nz)/2)])
sub_data <- leukemia.train[, c(var_nz, "V7130")]
# Gene expression data subset from patients with acute myeloid leukemia.
subset_1 <- subset(sub_data, sub_data$V7130==1)
subset_1 <- as.matrix(subset_1[, -dim(subset_1)[2]])
# The parameters of B and Boots in the following example are set as small values to
# reduce the running time, however the default values are proposed.
Sparse.graph.fit1 <- Sparse.graph(subset_1, graph.type=c("gaussian"),
                                  B=2, Boots=1, edge.rule=c("OR"))
# Give out the adjacency matrix of variables.
Sparse.graph.fit1$adj.matrix

# Example 2: Gaussian graph with AND-rule.
```

```
# The parameters of B and Boots in the following example are set as small values to
# reduce the running time, however the default values are proposed.
Sparse.graph.fit2 <- Sparse.graph(subset_1, graph.type=c("gaussian"),
                        B=2, Boots=1, edge.rule=c("OR"), plot=FALSE)
# Give out the adjacency matrix of variables.
Sparse.graph.fit2$adj.matrix
# Plot the graph based on the adjacency matrix of variables using the qgraph package.
library(qgraph)
qgraph(Sparse.graph.fit2$adj.matrix, directed=FALSE, color="blue",
        negCol="red", edge.labels=TRUE, layout="circle")
```

`TSLasso` *Two-stage hybrid LASSO model.*

Description

This function performs a LASSO logistic regression model using a two-stage hybrid procedure, namely the TSLasso logistic regression model, produces an optimal set of predictors and returns the robust estimations of coefficients of the selected predictors.

Usage

```
TSLasso(x, y, lambda.candidates = list(seq(0.001, 5, by = 0.01)), kfold = 10, seed = 0123)
```

Arguments

`x`	predictor matrix.
`y`	response variable, a factor object with values of 0 and 1.
`lambda.candidates`	the lambda candidates in the cv.lqa function, with the default values from 0.001 to 5 by=0.01.
`kfold`	the number of folds of cross validation - default is 10. Although kfold can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is kfold=3.
`seed`	the seed for random sampling, with the default value 0123.

Details

This function runs the LASSO logistic regression model using a two-stage hybrid procedure. In the two-stage hybrid penalized regression model, the LASSO algorithm is performed to obtain an initial estimator of the coefficients and to reduce the dimension of the model. The coefficient estimates of variables screened by the first stage are used for the weighting parameters of the adaptive LASSO in the second stage to select consistent variables. Accordingly, a portion of irrelevant variables are eliminated during the first stage and a relatively sparse set of variables is obtained. The glmnet algorithm is used for the LASSO estimation in the first stage and the optimal tuning parameter is selected via the K-fold cross-validation. The coefficients of the adaptive LASSO are estimated using the local quadratic approximation algorithm, which is proposed to approximate the nonconvex penalty function in generalized linear models based on penalized likelihood inference. Users can reduce the running time by using 3-fold CV, but the proposed 10-fold CV is assumed by default.

Value

var.selected	significant variables that are selected by the TSLasso model.
var.coef	coefficients of the selected significant variables.

References

[1] Guo, P., Zeng, F., Hu, X., Zhang, D., Zhu, S., Deng, Y., Hao, Y. (2015). Improved Variable Selection Algorithm Using a LASSO-Type Penalty, with an Application to Assessing Hepatitis B Infection Relevant Factors in Community Residents. PLoS One, 27;10(7):e0134151.

[2] Zou, H. (2006). The Adaptive Lasso And Its Oracle Properties. Journal of the American Statistical Association, 101(476), 1418:1429.

Examples

```
library(datasets)
head(iris)
X <- as.matrix(subset(iris, iris$Species!="virginica")[, -5])
Y <- as.numeric(ifelse(subset(iris,iris$Species!="virginica")[, 5]=='versicolor', 0, 1))
# Fit a two-stage hybrid LASSO (TSLasso) logistic regression model.
# The parameters of lambda.candidates in the following example are set as small values to
# reduce the running time, however the default values are proposed.
TSLasso.fit <- TSLasso(x=X, y=Y, lambda.candidates=list(seq(0.1, 1, by=0.05)),
                       kfold=3, seed=0123)
# Variables selected by the TSLasso model.
TSLasso.fit$var.selected
# Coefficients of the selected variables.
TSLasso.fit$var.coef
```

Index