Package ‘TraceAssist’

October 12, 2022

Type Package
Title Nonparametric Trace Regression via Sign Series Representation
Version 0.1.0
Author Chanwoo Lee <chanwoo.lee@wisc.edu>, Lexin Li <lexinli@berkeley.edu>, Hao Helen Zhang <hzhang@math.arizona.edu>, Miaoyan Wang <miaoyan.wang@wisc.edu>
Maintainer Chanwoo Lee <chanwoo.lee@wisc.edu>
Imports quadprog, Matrix, glmnet, keras
License GPL (>= 2)
Encoding UTF-8
RoxygenNote 7.1.1
NeedsCompilation no
Repository CRAN
Date/Publication 2021-05-10 08:42:14 UTC

R topics documented:

ADMM ................................................................. 2
CNN ................................................................. 3
Lasso ................................................................. 4
TraceAssist ......................................................... 4

Index 7
ADMM algorithm for weighted classification

Description

Implement an ADMM algorithm to optimize the weighted classification loss.

Usage

ADMM(X,ybar,Weight,Covariate=NULL,r,srow,scol,lambda=0,rho.ini=1)

Arguments

- **X**: A list of matrix-valued predictors.
- **ybar**: A vector of shifted response variables.
- **Weight**: Classification weight.
- **Covariate**: Additional covariates including intercept. `Covariate = NULL` indicates no covariates.
- **r**: The rank of coefficient matrix to be fitted.
- **srow**: The number of zero rows in coefficient matrix.
- **scol**: The number of zero columns in coefficient matrix.
- **lambda**: Lagrangian multiplier. Default is zero.
- **rho.ini**: Initial step size. Default is 1.

Value

The returned object is a list of components.
- **intercept**: The estimated intercept of the classifier.
- **P_row**: The left-singular vectors of the coefficient matrix.
- **P_col**: The right-singular vectors of the coefficient matrix.
- **obj**: Trajectory of weighted classification loss values over iterations.
- **iter**: The number of iterations.
- **fitted**: A vector of fitted response from estimated classifier.
- **B**: The estimated coefficient matrix of the classifier.

References

Examples

```r
#### Generate matrix predictors #######
X = list()
for(i in 1:10){
  X[[i]] = matrix(runif(4,-1,1),nrow = 2,ncol = 2)
}

#### Generate coefficient matrix #######
B = runif(2,-1,1)%*%t(runif(2,-1,1))

#### Generate response variables #######
y = NULL
for(i in 1:10){
  y = c(y,sign(sum(X[[i]]*B)+rnorm(1,sd = 0.1)))
}

#### classification with equal weights ######
res = ADMM(X,y,rep(1,10),r = 1,srow = 0,scol = 0)

#### Misclassification rate on training data ####
mean(sign(res$fitted)-y)
```

CNN

Convolutional Neural Network (CNN) with two hidden layers

Description

Implement a CNN with two hidden layers and ReLU activation.

Usage

```r
CNN(X,y,X_new,plot.figure = FALSE)
```

Arguments

- **X**: A list of matrix-valued predictors.
- **y**: Binary response variable.
- **X_new**: A list of new matrices in the test data.
- **plot.figure**: Option for plotting trajectory of accuracy over epochs.

Value

The returned object is a list of components.

- **prob**: The predicted probabilities for the test data.
- **class**: The estimated binary response for the test data.
- **history**: The trajectory of classification accuracy over epochs.
- **acc**: The classification accuracy on test data.
**Lasso**

*Logistic probability model via penalized maximum likelihood*

**Description**

Fit a logistic probability model based on Lasso penalty

**Usage**

\[
\text{Lasso}(xvec, y, xnew, \lambda)
\]

**Arguments**

- **xvec**: An input matrix. Each row is a vectorized predictor.
- **y**: Binary response variable.
- **xnew**: New predictors in the test data. Organized as a matrix with each row being a data point.
- **\lambda**: The regularization penalty.

**Value**

The returned object is a list of components.
- **B_est**: The estimated coefficient vector of linear predictor.
- **prob**: The predicted probabilities for the test data.

---

**TraceAssist**

*Aggregation of structured sign series for trace regression (ASSIST)*

**Description**

Main function for fitting the nonparametric trace regression. The algorithm uses a learning reduction approach to estimate the nonparametric trace regression via ASSIST.

**Usage**

\[
\text{TraceAssist}(X, y, X_{new=\text{NULL}}, r, \text{sparse}_r, \text{sparse}_c, H=10, \lambda=0, \rho_{ini}=0.1, \text{min}, \text{max})
\]
Arguments

- **X**: A list of matrix-valued predictors.
- **y**: A vector of response variables.
- **X_new**: A list of new matrices in the test data. If **X_new** is **NULL**, it returns fitted values in the training data.
- **r**: The rank of the sign representable function to be fitted.
- **sparse_r**: The number of zero rows in the coefficient matrix.
- **sparse_c**: The number of zero columns in the coefficient matrix.
- **H**: Resolution parameter that controls the number of classifiers to aggregate.
- **lambda**: Lagrangian multiplier.
- **rho.init**: Initial step size.
- **min**: Minimum value of the response variables.
- **max**: Maximum value of the response variables.

Value

The returned object is a list of components:

- **B_est**: An array that collects a series of coefficient matrices for the classifiers used in the algorithm.
- **fitted**: The predicted responses in the test data.
- **sign_fitted**: A matrix that collects a series of predicted signs for the classifiers used in the algorithm.

References


Examples

```r
######### Generate matrices in the training data ###########
X = list()
for(i in 1:10){
  X[[i]] = matrix(runif(4,-1,1),nrow = 2,ncol = 2)
}

######### Generate coefficient matrix ######################
B = runif(2,-1,1)%*%t(runif(2,-1,1))

######### Generate response variables #######################
y = NULL; signal = NULL
for(i in 1:10){
  signal = c(signal,sum(X[[i]]*B))
  y = c(y,sum(X[[i]]*B)+rnorm(1, sd = 0.1))
}
```
# Run ASSIST

```r
res = TraceAssist(X, y, r = 1, sparse_r = 0, sparse_c = 0, min = min(y), max = max(y))
mean(abs(res$fitted - signal))
```

# Generate new matrices in the test data

```r
X_new = list()
for (i in 1:10) {
  X_new[[i]] = matrix(runif(4, -1, 1), nrow = 2, ncol = 2)
}
```

# Generate response variables from X_new

```r
y_new = NULL
for (i in 1:10) {
  y_new = c(y_new, sum(X_new[[i]] * B))
}
```

# Run ASSIST

```r
res = TraceAssist(X, y, X_new, r = 1, sparse_r = 0, sparse_c = 0, min = min(y), max = max(y))
mean(abs(res$fitted - y_new))
```
Index

ADMM, 2
CNN, 3
Lasso, 4
TraceAssist, 4