Package ‘GAGAs’

December 9, 2022

Type Package

Title Global Adaptive Generative Adjustment Algorithm for Generalized Linear Models

Version 0.5.1

Language en-US

Description Fits linear regression, logistic and multinomial regression models, Poisson regression, Cox model via Global Adaptive Generative Adjustment Algorithm.
For more detailed information, see Bin Wang, Xiaofei Wang and Jianhua Guo (2022) <arXiv:1911.00658>.
This paper provides the theoretical properties of Gaga linear model when the load matrix is orthogonal.
Further study is going on for the nonorthogonal cases and generalized linear models.
These works are in part supported by the National Natural Foundation of China (No.12171076).

License GPL-2

URL https://arxiv.org/abs/1911.00658

Encoding UTF-8

Depends R (>= 3.6.0)

Imports Rcpp (>= 1.0.9), survival

Suggests mvtnorm

SystemRequirements C++14

LinkingTo Rcpp, RcppEigen

RoxygenNote 7.2.2

Maintainer Bin Wang <eatingbeen@hotmail.com>

NeedsCompilation yes

Author Bin Wang [aut, cre],
Xiaofei Wang [ctb],
Jianhua Guo [ths]

Repository CRAN

Date/Publication 2022-12-08 23:52:30 UTC

R topics documented:

`cal.acc` *Calculate ACC for classification, the inputs must be characters*

Description

Calculate ACC for classification, the inputs must be characters

Usage

```
cal.acc(predictions, truelabels)
```

Arguments

`predictions`	predictions
`truelabels`	true labels

Value

ACC

Examples

```
set.seed(2022)
p_size = 30
sample_size=300
R1 = 3
R2 = 2
rate = 0.5 #Proportion of value zero in beta
# Set true beta
zeroNum = round(rate*p_size)
ind = sample(1:p_size,zeroNum)
beta_true = runif(p_size,0,R2)
beta_true[ind] = 0
X = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
y=X%*%beta_true + rnorm(sample_size,mean=0,sd=2)
# Estimation
fit = GAGAs(X,y,alpha = 3,family="gaussian")
Eb = fit$beta
#Create testing data
X_t = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
y_t=X_t%*%beta_true + rnorm(sample_size,mean=0,sd=2)
#Prediction
Ey = predict.GAGA(fit,newx=X_t)

cat("\n err:", norm(Eb-beta_true,type="2")/norm(beta_true,type="2"))
cat("\n acc:", cal.acc(as.character(Eb!=0),as.character(beta_true!=0)))
cat("\n perr:", norm(Ey-y_t,type="2")/sqrt(sample_size))
```

`cal.cindex` *compute C index for a Cox model*

Description

Computes Harrel's C index for predictions from a `"cox"` object.

Usage

```
cal.cindex(pred, y, weights = rep(1, nrow(y)))
```

Arguments

`pred`	Predictions from a `"cox"` object
`y`	a survival response object - a matrix with two columns "time" and "status"; see documentation for "glmnet" or see documentation for "GAGA"
`weights`	optional observation weights

Details

Computes the concordance index, taking into account censoring. This file fully references the Cindex.R file in glmnet package.

Value

Harrel's C index

Author(s)

Trevor Hastie <hastie@stanford.edu>

References

Harrel Jr, F. E. and Lee, K. L. and Mark, D. B. (1996) *Tutorial in biostatistics: multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing error*, Statistics in Medicine, 15, pages 361–387.#'

See Also

`cv.glmnet`

Examples

```
set.seed(10101)
N = 1000
p = 30
nzc = p/3
x = matrix(rnorm(N * p), N, p)
beta = rnorm(nzc)
fx = x[, seq(nzc)] %*% beta/3
hx = exp(fx)
ty = rexp(N, hx)
tcens = rbinom(n = N, prob = 0.3, size = 1)  # censoring indicator
y = cbind(time = ty, status = 1 - tcens)  # y=Surv(ty,1-tcens) with library(survival)
fit = GAGAs(x, y, family = "cox")
pred = predict(fit, newx = x)
cat("\n Cindex:", cal.cindex(pred, y))
```

`cal.w.acc` *Calculate the weighted ACC of the classification, the inputs must be characters*

Description

Calculate the weighted ACC of the classification, the inputs must be characters

Usage

```
cal.w.acc(predictions, truelabels)
```

Arguments

`predictions`	predictions
`truelabels`	true labels

Value

weighted ACC

Examples

```
set.seed(2022)
p_size = 30
sample_size=300
R1 = 3
R2 = 2
rate = 0.5 #Proportion of value zero in beta
# Set true beta
zeroNum = round(rate*p_size)
ind = sample(1:p_size,zeroNum)
beta_true = runif(p_size,0,R2)
beta_true[ind] = 0
X = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
y=X%*%beta_true + rnorm(sample_size,mean=0,sd=2)
# Estimation
fit = GAGAs(X,y,alpha = 3,family="gaussian")
Eb = fit$beta
#Create testing data
X_t = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
y_t=X_t%*%beta_true + rnorm(sample_size,mean=0,sd=2)
#Prediction
Ey = predict.GAGA(fit,newx=X_t)

cat("\n err:", norm(Eb-beta_true,type="2")/norm(beta_true,type="2"))
cat("\n acc:", cal.w.acc(as.character(Eb!=0),as.character(beta_true!=0)))
cat("\n perr:", norm(Ey-y_t,type="2")/sqrt(sample_size))
```

`cox_GAGA` *Fit a Cox model via the GAGA algorithm.*

Description

Fit a Cox model via the Global Adaptive Generative Adjustment algorithm. Part of this function refers to the coxphfit function in MATLAB 2016b.

Usage

```
cox_GAGA(
  X,
  t,
  alpha = 2,
  itrNum = 20,
  thresh = 0.001,
  flag = TRUE,
  lamda_0 = 0.5,
  fdiag = TRUE
)
```

Arguments

`X`	Input matrix, of dimension nobs*nvars; each row is an observation. If the intercept term needs to be considered in the estimation process, then the first column of `X` must be all 1s.
`t`	A n*2 matrix, one column should be named "time", indicating the survival time; the other column must be named "status", and consists of 0 and 1, 0 indicates that the row of data is censored, 1 is opposite.
`alpha`	Hyperparameter. The suggested value for alpha is 2 or 3.
`itrNum`	The number of iteration steps. In general, 20 steps are enough. If the condition number of `X` is large, it is recommended to greatly increase the number of iteration steps.
`thresh`	Convergence threshold for beta Change, if `max(abs(beta-beta_old))<threshold`, return.
`flag`	It identifies whether to make model selection. The default is `TRUE`.
`lamda_0`	The initial value of the regularization parameter for ridge regression. The running result of the algorithm is not sensitive to this value.
`fdiag`	It identifies whether to use diag Approximation to speed up the algorithm.

Value

Coefficient vector.

Examples

```
set.seed(2022)
p_size = 50
sample_size = 500
test_size = 1000
R1 = 3
R2 = 1
rate = 0.5 #Proportion of value zero in beta
censoringRate = 0.25 #Proportion of censoring data in observation data
# Set true beta
zeroNum = round(rate*p_size)
```

```
ind = sample(1:p_size,zeroNum)#'
beta_true = runif(p_size,-R2,R2)
beta_true[ind] = 0
# Generate training samples
X = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
z = X%*%beta_true
u = runif(sample_size,0,1)
t = ((-log(1-u)/(3*exp(z)))*100)^(0.1)
cs = rep(0,sample_size)
csNum = round(censoringRate*sample_size)
ind = sample(1:sample_size,csNum)#'
cs[ind] = 1
t[ind] = runif(csNum,0,0.8)*t[ind]
y = cbind(t,1 - cs)
colnames(y) = c("time", "status")
#Estimation
fit = GAGAs(X,y,alpha=2,family="cox")
Eb = fit$beta

#Generate testing samples
X_t = R1*matrix(rnorm(test_size * p_size), ncol = p_size)
z = X_t%*%beta_true
u = runif(test_size,0,1)
t = ((-log(1-u)/(3*exp(z)))*100)^(0.1)
cs = rep(0,test_size)
csNum = round(censoringRate*test_size)
ind = sample(1:test_size,csNum)#'
cs[ind] = 1
t[ind] = runif(csNum,0,0.8)*t[ind]
y_t = cbind(t,1 - cs)
colnames(y_t) = c("time", "status")
#Prediction
pred = predict(fit,newx=X_t)

cat("\n err:", norm(Eb-beta_true,type="2")/norm(beta_true,type="2"))
cat("\n acc:", cal.w.acc(as.character(Eb!=0),as.character(beta_true!=0)))
cat("\n Cindex:", cal.cindex(pred,y_t))
```

cpp_COX_gaga — *Fit a Cox model via the GAGA algorithm using cpp.*

Description

Fit a Cox model via the Global Adaptive Generative Adjustment algorithm. Part of this function refers to the coxphfit function in MATLAB 2016b.

Usage

```
cpp_COX_gaga(
  X,
```

```
    y,
    cens,
    alpha = 2,
    itrNum = 50L,
    thresh = 0.001,
    flag = TRUE,
    lamda_0 = 0.5,
    fdiag = TRUE
  )
```

Arguments

`X`	Input matrix, of dimension nobs*nvars; each row is an observation. If the intercept term needs to be considered in the estimation process, then the first column of `X` must be all 1s.
`y`	A n*1 matrix, indicating the survival time;
`cens`	A n*1 matrix, consists of 0 and 1, 1 indicates that the row of data is censored, 0 is opposite.
`alpha`	Hyperparameter. The suggested value for alpha is 2 or 3.
`itrNum`	The number of iteration steps. In general, 20 steps are enough.
`thresh`	Convergence threshold for beta Change, if `max(abs(beta-beta_old))<threshold`, return.
`flag`	It identifies whether to make model selection. The default is `TRUE`.
`lamda_0`	The initial value of the regularization parameter for ridge regression.
`fdiag`	It identifies whether to use diag Approximation to speed up the algorithm.

Value

Coefficient vector

`cpp_logistic_gaga` *Fit a logistic model via the Global Adaptive Generative Adjustment algorithm using cpp*

Description

Fit a logistic model via the Global Adaptive Generative Adjustment algorithm using cpp

Usage

```
cpp_logistic_gaga(
  X,
  y,
  s_alpha,
  s_itrNum,
```

```
  s_thresh,
  s_flag,
  s_lamda_0,
  s_fdiag
)
```

Arguments

X	Input matrix, of dimension nobs*nvars; each row is an observation. If the intercept term needs to be considered in the estimation process, then the first column of X must be all 1s.
y	should be either a factor with two levels.
s_alpha	Hyperparameter. The suggested value for alpha is 1 or 2. When the collinearity of the load matrix is serious, the hyperparameters can be selected larger, such as 5.
s_itrNum	The number of iteration steps. In general, 20 steps are enough. If the condition number of X is large, it is recommended to greatly increase the number of iteration steps.
s_thresh	Convergence threshold for beta Change, if max(abs(beta-beta_old))<threshold, return.
s_flag	It identifies whether to make model selection. The default is TRUE.
s_lamda_0	The initial value of the regularization parameter for ridge regression. The running result of the algorithm is not sensitive to this value.
s_fdiag	It identifies whether to use diag Approximation to speed up the algorithm.

Value

Coefficient vector.

---

cpp_multinomial_gaga *Fit a multinomial model via the GAGA algorithm using cpp*

Description

Fit a multinomial model the Global Adaptive Generative Adjustment algorithm

Usage

```
cpp_multinomial_gaga(
  X,
  y,
  s_alpha,
  s_itrNum,
  s_thresh,
  s_flag,
  s_lamda_0,
  s_fdiag
)
```

Arguments

X	Input matrix, of dimension nobs*nvars; each row is an observation. If the intercept term needs to be considered in the estimation process, then the first column of X must be all 1s.
y	a One-hot response matrix or a nc>=2 level factor
s_alpha	Hyperparameter. The suggested value for alpha is 1 or 2. When the collinearity of the load matrix is serious, the hyperparameters can be selected larger, such as 5.
s_itrNum	The number of iteration steps. In general, 20 steps are enough. If the condition number of X is large, it is recommended to greatly increase the number of iteration steps.
s_thresh	Convergence threshold for beta Change, if max(abs(beta-beta_old))<threshold, return.
s_flag	It identifies whether to make model selection. The default is TRUE.
s_lamda_0	The initial value of the regularization parameter for ridge regression. The running result of the algorithm is not sensitive to this value.
s_fdiag	It identifies whether to use diag Approximation to speed up the algorithm.

Value

Coefficient matrix with K-1 columns beta_1,...,beta_K-1 where K is the class number. For k=1,..,K-1, the probability

$$Pr(G = k|x) = exp(x^T beta_k)/(1 + sum_{k=1}^{K-1} exp(x^T beta_k))$$

. For k=K, the probability

$$Pr(G = K|x) = 1/(1 + sum_{k=1}^{K-1} exp(x^T beta_k))$$

.

---

cpp_poisson_gaga *Fit a poisson model via the GAGA algorithm using cpp*

Description

Fit a poisson model the Global Adaptive Generative Adjustment algorithm

Usage

```
cpp_poisson_gaga(X, y, s_alpha, s_itrNum, s_thresh, s_flag, s_lamda_0, s_fdiag)
```

Arguments

- `X` Input matrix, of dimension nobs*nvars; each row is an observation. If the intercept term needs to be considered in the estimation process, then the first column of `X` must be all 1s. In order to run the program stably, it is recommended that the value of X should not be too large. It is recommended to preprocess all the items in X except the intercept item by means of preprocessing, so that the mean value of each column is 0 and the standard deviation is `1/ colnum(X)`.
- `y` Non-negative count response vector.
- `s_alpha` Hyperparameter. The suggested value for alpha is 1 or 2. When the collinearity of the load matrix is serious, the hyperparameters can be selected larger, such as 5.
- `s_itrNum` The number of iteration steps. In general, 20 steps are enough. If the condition number of `X` is large, it is recommended to greatly increase the number of iteration steps.
- `s_thresh` Convergence threshold for beta Change, if `max(abs(beta-beta_old))<threshold`, return.
- `s_flag` It identifies whether to make model selection. The default is `TRUE`.
- `s_lamda_0` The initial value of the regularization parameter for ridge regression. The running result of the algorithm is not sensitive to this value.
- `s_fdiag` It identifies whether to use diag Approximation to speed up the algorithm.

Value

Coefficient vector.

`GAGAs` *GAGAs: A package for fiting a GLM with GAGA algorithm*

Description

Fits linear, logistic and multinomial, poisson, and Cox regression models via Global Adaptive Generative Adjustment algorithm.

Fits linear, logistic and multinomial, poisson, and Cox regression models via Global Adaptive Generative Adjustment algorithm.

Usage

```
GAGAs(
  X,
  y,
  family = c("gaussian", "binomial", "poisson", "multinomial", "cox"),
  alpha = 2,
  itrNum = 500,
  thresh = 0.001,
```

```
  QR_flag = FALSE,
  flag = TRUE,
  lamda_0 = 0.001,
  fdiag = TRUE,
  frp = TRUE
)
```

Arguments

`X`	Input matrix, of dimension nobs*nvars; each row is an observation. If the intercept term needs to be considered in the estimation process, then the first column of `X` must be all 1s.
`y`	Response variable. Quantitative for `family="gaussian"`, or `family="poisson"` (non-negative counts). For `family="binomial"` should be either a factor with two levels. For `family="multinomial"` should be a one-hot matrix or a `nc>=2` level factor. For `family="cox"` should be an n*2 matrix, one column should be named "time", indicating the survival time; the other column must be named "status", and consists of 0 and 1, 0 indicates that the row of data is censored, 1 is opposite.
`family`	Either a character string representing one of the built-in families, `"gaussian"`, `"binomial"`, `"poisson"`, `"multinomial"` or `"cox"`.
`alpha`	Hyperparameter. In general, alpha can be set to 1, 2 or 3. for `family="gaussian"` and `family="cox"`, the suggested value for alpha is 2 or 3. for `family="binomial"`, `family="poisson"` and `family="multinomial"`, the suggested value for alpha is 1 or 2. but when the collinearity of the load matrix is serious, the hyperparameters can be selected larger, such as 5.
`itrNum`	The number of iteration steps. In general, 20 steps are enough. If the condition number of `X` is large, it is recommended to greatly increase the number of iteration steps.
`thresh`	Convergence threshold for beta Change, if `max(abs(beta-beta_old))<threshold`, return.
`QR_flag`	It identifies whether to use QR decomposition to speed up the algorithm. Currently only valid for linear models.
`flag`	It identifies whether to make model selection. The default is `TRUE`.
`lamda_0`	The initial value of the regularization parameter for ridge regression. The running result of the algorithm is not sensitive to this value.
`fdiag`	It identifies whether to use diag Approximation to speed up the algorithm.
`frp`	Identifies if a method is preprocessed to reduce the number of parameters

Value

Regression coefficients

Mypackage functions

GAGAs

Examples

```
# Gaussian
set.seed(2022)
p_size = 30
sample_size=300
R1 = 3
R2 = 2
rate = 0.5 #Proportion of value zero in beta
# Set true beta
zeroNum = round(rate*p_size)
ind = sample(1:p_size,zeroNum)
beta_true = runif(p_size,0,R2)
beta_true[ind] = 0
X = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
y=X%*%beta_true + rnorm(sample_size,mean=0,sd=2)
# Estimation
fit = GAGAs(X,y,alpha = 3,family="gaussian")
Eb = fit$beta
#Create testing data
X_t = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
y_t=X_t%*%beta_true + rnorm(sample_size,mean=0,sd=2)
#Prediction
Ey = predict.GAGA(fit,newx=X_t)

cat("\n err:", norm(Eb-beta_true,type="2")/norm(beta_true,type="2"))
cat("\n acc:", cal.w.acc(as.character(Eb!=0),as.character(beta_true!=0)))
cat("\n perr:", norm(Ey-y_t,type="2")/sqrt(sample_size))

# binomial
set.seed(2022)
cat("\n")
cat("Test binomial GAGA\n")
p_size = 30
sample_size=600
test_size=1000
R1 = 1
R2 = 3
rate = 0.5 #Proportion of value zero in beta
#Set true beta
zeroNum = round(rate*p_size)
ind = sample(1:p_size,zeroNum)
beta_true = runif(p_size,R2*0.2,R2)
beta_true[ind] = 0
X = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
X[1:sample_size,1]=1
t = 1/(1+exp(-X%*%beta_true))
tmp = runif(sample_size,0,1)
y = rep(0,sample_size)
y[t>tmp] = 1
fit = GAGAs(X,y,family = "binomial", alpha = 1)
Eb = fit$beta
#Generate test samples
```

```
X_t = R1*matrix(rnorm(test_size * p_size), ncol = p_size)
X_t[1:test_size,1]=1
t = 1/(1+exp(-X_t%*%beta_true))
tmp = runif(test_size,0,1)
y_t = rep(0,test_size)
y_t[t>tmp] = 1
#Prediction
Ey = predict(fit,newx = X_t)
cat("\n--------------------")
cat("\n err:", norm(Eb-beta_true,type="2")/norm(beta_true,type="2"))
cat("\n acc:", cal.w.acc(as.character(Eb!=0),as.character(beta_true!=0)))
cat("\n pacc:", cal.w.acc(as.character(Ey),as.character(y_t)))
cat("\n")
# multinomial
set.seed(2022)
cat("\n")
cat("Test multinomial GAGA\n")
p_size = 20
C = 3
classnames = c("C1","C2","C3","C4")
sample_size = 500
test_size = 1000
rate = 0.5 #Proportion of value zero in beta
Num = 10 # Total number of experiments
R1 = 1
R2 = 5
#Set true beta
beta_true = matrix(rep(0,p_size*C),c(p_size,C))
zeroNum = round(rate*p_size)
for(jj in 1:C){
  ind = sample(1:p_size,zeroNum)
  tmp = runif(p_size,0,R2)
  tmp[ind] = 0
  beta_true[,jj] = tmp
}
#Generate training samples
X = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
X[1:sample_size,1]=1
z = X%*%beta_true
t = exp(z)/(1+rowSums(exp(z)))
t = cbind(t,1-rowSums(t))
tt = t(apply(t,1,cumsum))
tt = cbind(rep(0,sample_size),tt)
# y = matrix(rep(0,sample_size*(C+1)),c(sample_size,C+1))
y = rep(0,sample_size)
for(jj in 1:sample_size){
  tmp = runif(1,0,1)
  for(kk in 1:(C+1)){
    if((tmp>tt[jj,kk])&&(tmp<=tt[jj,kk+1])){
      # y[jj,kk] = 1
      y[jj] = kk
      break
    }
```

```
  }
}
y = classnames[y]
fit = GAGAs(X, y,alpha=1,family = "multinomial")
Eb = fit$beta
#Prediction
#Generate test samples
X_t = R1*matrix(rnorm(test_size * p_size), ncol = p_size)
X_t[1:test_size,1]=1
z = X_t%*%beta_true
t = exp(z)/(1+rowSums(exp(z)))
t = cbind(t,1-rowSums(t))
tt = t(apply(t,1,cumsum))
tt = cbind(rep(0,test_size),tt)
y_t = rep(0,test_size)
for(jj in 1:test_size){
  tmp = runif(1,0,1)
  for(kk in 1:(C+1)){
    if((tmp>tt[jj,kk])&&(tmp<=tt[jj,kk+1])){
      y_t[jj] = kk
      break
    }
  }
}
y_t = classnames[y_t]
Ey = predict(fit,newx = X_t)
cat("\n--------------------")
cat("\n err:", norm(Eb-beta_true,type="2")/norm(beta_true,type="2"))
cat("\n acc:", cal.w.acc(as.character(Eb!=0),as.character(beta_true!=0)))
cat("\n pacc:", cal.w.acc(as.character(Ey),as.character(y_t)))
cat("\n")

# Poisson
set.seed(2022)
p_size = 30
sample_size=300
R1 = 1/sqrt(p_size)
R2 = 5
rate = 0.5 #Proportion of value zero in beta
# Set true beta
zeroNum = round(rate*p_size)
ind = sample(1:p_size,zeroNum)#'
beta_true = runif(p_size,0,R2)
beta_true[ind] = 0
X = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
X[1:sample_size,1]=1
y = rpois(sample_size,lambda = as.vector(exp(X%*%beta_true)))
y = as.vector(y)
# Estimate
fit = GAGAs(X,y,alpha = 2,family="poisson")
Eb = fit$beta
cat("\n err:", norm(Eb-beta_true,type="2")/norm(beta_true,type="2"))
cat("\n acc:", cal.w.acc(as.character(Eb!=0),as.character(beta_true!=0)))
```

```
# cox
p_size = 50
sample_size = 500
test_size = 1000
R1 = 3
R2 = 1
rate = 0.5 #Proportion of value zero in beta
censoringRate = 0.25 #Proportion of censoring data in observation data
# Set true beta
zeroNum = round(rate*p_size)
ind = sample(1:p_size,zeroNum)#'
beta_true = runif(p_size,-R2,R2)
beta_true[ind] = 0
# Generate training samples
X = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
z = X%*%beta_true
u = runif(sample_size,0,1)
t = ((-log(1-u)/(3*exp(z)))*100)^(0.1)
cs = rep(0,sample_size)
csNum = round(censoringRate*sample_size)
ind = sample(1:sample_size,csNum)#'
cs[ind] = 1
t[ind] = runif(csNum,0,0.8)*t[ind]
y = cbind(t,1 - cs)
colnames(y) = c("time", "status")
#Estimation
fit = GAGAs(X,y,alpha=2,family="cox")
Eb = fit$beta

#Generate testing samples
X_t = R1*matrix(rnorm(test_size * p_size), ncol = p_size)
z = X_t%*%beta_true
u = runif(test_size,0,1)
t = ((-log(1-u)/(3*exp(z)))*100)^(0.1)
cs = rep(0,test_size)
csNum = round(censoringRate*test_size)
ind = sample(1:test_size,csNum)#'
cs[ind] = 1
t[ind] = runif(csNum,0,0.8)*t[ind]
y_t = cbind(t,1 - cs)
colnames(y_t) = c("time", "status")
#Prediction
pred = predict(fit,newx=X_t)

cat("\n err:", norm(Eb-beta_true,type="2")/norm(beta_true,type="2"))
cat("\n acc:", cal.w.acc(as.character(Eb!=0),as.character(beta_true!=0)))
cat("\n Cindex:", cal.cindex(pred,y_t))
```

LM_GAGA *Fit a linear model via the GAGA algorithm*

Description

Fit a linear model with a Gaussian noise via the Global Adaptive Generative Adjustment algorithm

Usage

```
LM_GAGA(
  X,
  y,
  alpha = 3,
  itrNum = 50,
  thresh = 0.001,
  QR_flag = FALSE,
  flag = TRUE,
  lamda_0 = 0.001,
  fix_sigma = FALSE,
  sigm2_0 = 1,
  fdiag = TRUE,
  frp = TRUE
)
```

Arguments

`X`	Input matrix, of dimension nobs*nvars; each row is an observation. If the intercept term needs to be considered in the estimation process, then the first column of `X` must be all 1s.
`y`	Quantitative response vector.
`alpha`	Hyperparameter. The suggested value for alpha is 2 or 3. When the collinearity of the load matrix is serious, the hyperparameters can be selected larger, such as 5.
`itrNum`	The number of iteration steps. In general, 20 steps are enough. If the condition number of `X` is large, it is recommended to greatly increase the number of iteration steps.
`thresh`	Convergence threshold for beta Change, if `max(abs(beta-beta_old))<threshold`, return.
`QR_flag`	It identifies whether to use QR decomposition to speed up the algorithm. Currently only valid for linear models.
`flag`	It identifies whether to make model selection. The default is `TRUE`.
`lamda_0`	The initial value of the regularization parameter for ridge regression. The running result of the algorithm is not sensitive to this value.
`fix_sigma`	It identifies whether to update the variance estimate of the Gaussian noise or not. `fix_sigma=TRUE` uses the initial variance as the variance estimate in each loop. `fix_sigma=FALSE` updates the variance estimate in each loop.
`sigm2_0`	The initial variance of the Gaussian noise.
`fdiag`	It identifies whether to use diag Approximation to speed up the algorithm.
`frp`	Identifies whether pre-processing is performed by the OMP method to reduce the number of parameters

Value

Coefficient vector.

Examples

```
# Gaussian
set.seed(2022)
p_size = 30
sample_size=300
R1 = 3
R2 = 2
rate = 0.5 #Proportion of value zero in beta
# Set true beta
zeroNum = round(rate*p_size)
ind = sample(1:p_size,zeroNum)
beta_true = runif(p_size,0,R2)
beta_true[ind] = 0
X = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
y=X%*%beta_true + rnorm(sample_size,mean=0,sd=2)
# Estimation
fit = GAGAs(X,y,alpha = 3,family="gaussian")
Eb = fit$beta
#Create testing data
X_t = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
y_t=X_t%*%beta_true + rnorm(sample_size,mean=0,sd=2)
#Prediction
Ey = predict.GAGA(fit,newx=X_t)

cat("\n err:", norm(Eb-beta_true,type="2")/norm(beta_true,type="2"))
cat("\n acc:", cal.w.acc(as.character(Eb!=0),as.character(beta_true!=0)))
cat("\n perr:", norm(Ey-y_t,type="2")/sqrt(sample_size))
```

---

logistic_GAGA — *Fit a logistic model via the Global Adaptive Generative Adjustment algorithm*

Description

Fit a logistic model via the Global Adaptive Generative Adjustment algorithm

Usage

```
logistic_GAGA(
  X,
  y,
  alpha = 1,
  itrNum = 30,
  thresh = 0.001,
  flag = TRUE,
```

```
  lamda_0 = 0.001,
  fdiag = TRUE
)
```

Arguments

- `X` — Input matrix, of dimension nobs*nvars; each row is an observation. If the intercept term needs to be considered in the estimation process, then the first column of `X` must be all 1s.
- `y` — should be either a factor with two levels.
- `alpha` — Hyperparameter. The suggested value for alpha is 1 or 2. When the collinearity of the load matrix is serious, the hyperparameters can be selected larger, such as 5.
- `itrNum` — The number of iteration steps. In general, 20 steps are enough. If the condition number of `X` is large, it is recommended to greatly increase the number of iteration steps.
- `thresh` — Convergence threshold for beta Change, if `max(abs(beta-beta_old))<threshold`, return.
- `flag` — It identifies whether to make model selection. The default is `TRUE`.
- `lamda_0` — The initial value of the regularization parameter for ridge regression. The running result of the algorithm is not sensitive to this value.
- `fdiag` — It identifies whether to use diag Approximation to speed up the algorithm.

Value

Coefficient vector.

Examples

```
# binomial
set.seed(2022)
cat("\n")
cat("Test binomial GAGA\n")
p_size = 30
sample_size=600
test_size=1000
R1 = 1
R2 = 3
rate = 0.5 #Proportion of value zero in beta
#Set true beta
zeroNum = round(rate*p_size)
ind = sample(1:p_size,zeroNum)
beta_true = runif(p_size,R2*0.2,R2)
beta_true[ind] = 0
X = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
X[1:sample_size,1]=1
t = 1/(1+exp(-X%*%beta_true))
tmp = runif(sample_size,0,1)
y = rep(0,sample_size)
```

```
y[t>tmp] = 1
fit = GAGAs(X,y,family = "binomial", alpha = 1)
Eb = fit$beta
#Generate test samples
X_t = R1*matrix(rnorm(test_size * p_size), ncol = p_size)
X_t[1:test_size,1]=1
t = 1/(1+exp(-X_t%*%beta_true))
tmp = runif(test_size,0,1)
y_t = rep(0,test_size)
y_t[t>tmp] = 1
#Prediction
Ey = predict(fit,newx = X_t)
cat("\n--------------------")
cat("\n err:", norm(Eb-beta_true,type="2")/norm(beta_true,type="2"))
cat("\n acc:", cal.w.acc(as.character(Eb!=0),as.character(beta_true!=0)))
cat("\n pacc:", cal.w.acc(as.character(Ey),as.character(y_t)))
cat("\n")
```

---

multinomial_GAGA *Fit a multinomial model via the GAGA algorithm*

Description

Fit a multinomial model the Global Adaptive Generative Adjustment algorithm

Usage

```
multinomial_GAGA(
  X,
  y,
  alpha = 1,
  itrNum = 50,
  thresh = 0.001,
  flag = TRUE,
  lamda_0 = 0.001,
  fdiag = TRUE
)
```

Arguments

X	Input matrix, of dimension nobs*nvars; each row is an observation. If the intercept term needs to be considered in the estimation process, then the first column of X must be all 1s.
y	a One-hot response matrix or a nc>=2 level factor
alpha	Hyperparameter. The suggested value for alpha is 1 or 2. When the collinearity of the load matrix is serious, the hyperparameters can be selected larger, such as 5.

itrNum	The number of iteration steps. In general, 20 steps are enough. If the condition number of X is large, it is recommended to greatly increase the number of iteration steps.
thresh	Convergence threshold for beta Change, if max(abs(beta-beta_old))<threshold, return.
flag	It identifies whether to make model selection. The default is TRUE.
lamda_0	The initial value of the regularization parameter for ridge regression. The running result of the algorithm is not sensitive to this value.
fdiag	It identifies whether to use diag Approximation to speed up the algorithm.

Value

Coefficient matrix with K-1 columns beta_1,...,beta_K-1 where K is the class number. For k=1,..,K-1, the probability

$$Pr(G = k|x) = exp(x^T beta_k)/(1 + sum_{k=1}^{K-1} exp(x^T beta_k))$$

. For k=K, the probability

$$Pr(G = K|x) = 1/(1 + sum_{k=1}^{K-1} exp(x^T beta_k))$$

.

Examples

```
# multinomial
set.seed(2022)
cat("\n")
cat("Test multinomial GAGA\n")
p_size = 20
C = 3
classnames = c("C1","C2","C3","C4")
sample_size = 500
test_size = 1000
rate = 0.5 #Proportion of value zero in beta
Num = 10 # Total number of experiments
R1 = 1
R2 = 5
#Set true beta
beta_true = matrix(rep(0,p_size*C),c(p_size,C))
zeroNum = round(rate*p_size)
for(jj in 1:C){
  ind = sample(1:p_size,zeroNum)
  tmp = runif(p_size,0,R2)
  tmp[ind] = 0
  beta_true[,jj] = tmp
}
#Generate training samples
X = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
X[1:sample_size,1]=1
z = X%*%beta_true
```

```
t = exp(z)/(1+rowSums(exp(z)))
t = cbind(t,1-rowSums(t))
tt = t(apply(t,1,cumsum))
tt = cbind(rep(0,sample_size),tt)
# y = matrix(rep(0,sample_size*(C+1)),c(sample_size,C+1))
y = rep(0,sample_size)
for(jj in 1:sample_size){
  tmp = runif(1,0,1)
  for(kk in 1:(C+1)){
    if((tmp>tt[jj,kk])&&(tmp<=tt[jj,kk+1])){
      # y[jj,kk] = 1
      y[jj] = kk
      break
    }
  }
}
y = classnames[y]
fit = GAGAs(X, y,alpha=1,family = "multinomial")
Eb = fit$beta
#Prediction
#Generate test samples
X_t = R1*matrix(rnorm(test_size * p_size), ncol = p_size)
X_t[1:test_size,1]=1
z = X_t%*%beta_true
t = exp(z)/(1+rowSums(exp(z)))
t = cbind(t,1-rowSums(t))
tt = t(apply(t,1,cumsum))
tt = cbind(rep(0,test_size),tt)
y_t = rep(0,test_size)
for(jj in 1:test_size){
  tmp = runif(1,0,1)
  for(kk in 1:(C+1)){
    if((tmp>tt[jj,kk])&&(tmp<=tt[jj,kk+1])){
      y_t[jj] = kk
      break
    }
  }
}
y_t = classnames[y_t]
Ey = predict(fit,newx = X_t)
cat("\n--------------------")
cat("\n err:", norm(Eb-beta_true,type="2")/norm(beta_true,type="2"))
cat("\n acc:", cal.w.acc(as.character(Eb!=0),as.character(beta_true!=0)))
cat("\n pacc:", cal.w.acc(as.character(Ey),as.character(y_t)))
cat("\n")
```

---

poisson_GAGA *Fit a Poisson model via the GAGA algorithm*

Description

Fit a Poisson model the Global Adaptive Generative Adjustment algorithm

Usage

```
poisson_GAGA(
  X,
  y,
  alpha = 1,
  itrNum = 30,
  thresh = 0.001,
  flag = TRUE,
  lamda_0 = 0.5,
  fdiag = TRUE
)
```

Arguments

`X`	Input matrix, of dimension nobs*nvars; each row is an observation. If the intercept term needs to be considered in the estimation process, then the first column of `X` must be all 1s. In order to run the program stably, it is recommended that the value of X should not be too large. It is recommended to preprocess all the items in X except the intercept item by means of preprocessing, so that the mean value of each column is 0 and the standard deviation is `1/ colnum(X)`.
`y`	Non-negative count response vector.
`alpha`	Hyperparameter. The suggested value for alpha is 1 or 2. When the collinearity of the load matrix is serious, the hyperparameters can be selected larger, such as 5.
`itrNum`	The number of iteration steps. In general, 20 steps are enough. If the condition number of `X` is large, it is recommended to greatly increase the number of iteration steps.
`thresh`	Convergence threshold for beta Change, if `max(abs(beta-beta_old))<threshold`, return.
`flag`	It identifies whether to make model selection. The default is `TRUE`.
`lamda_0`	The initial value of the regularization parameter for ridge regression. The running result of the algorithm is not sensitive to this value.
`fdiag`	It identifies whether to use diag Approximation to speed up the algorithm.

Value

Coefficient vector.

Examples

```
# Poisson
set.seed(2022)
p_size = 30
sample_size=300
R1 = 1/sqrt(p_size)
R2 = 5
rate = 0.5 #Proportion of value zero in beta
```

```
# Set true beta
zeroNum = round(rate*p_size)
ind = sample(1:p_size,zeroNum)#'
beta_true = runif(p_size,0,R2)
beta_true[ind] = 0
X = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
X[1:sample_size,1]=1
y = rpois(sample_size,lambda = as.vector(exp(X%*%beta_true)))
y = as.vector(y)
# Estimate
fit = GAGAs(X,y,alpha = 2,family="poisson")
Eb = fit$beta
cat("\n err:", norm(Eb-beta_true,type="2")/norm(beta_true,type="2"))
cat("\n acc:", cal.w.acc(as.character(Eb!=0),as.character(beta_true!=0)))
```

`predict.GAGA` *Get predictions from a GAGA fit object*

Description

Gives fitted values from a fitted GAGA object.

Usage

```
## S3 method for class 'GAGA'
predict(object, newx, ...)
```

Arguments

`object`	Fitted "GAGA" object.
`newx`	Matrix of new values for x at which predictions are to be made. Must be a matrix. If the intercept term needs to be considered in the estimation process, then the first column of `X` must be all 1s.
`...`	some other params

Value

Predictions

Examples

```
set.seed(2022)
p_size = 30
sample_size=300
R1 = 3
R2 = 2
rate = 0.5 #Proportion of value zero in beta
```

```
# Set true beta
zeroNum = round(rate*p_size)
ind = sample(1:p_size,zeroNum)
beta_true = runif(p_size,0,R2)
beta_true[ind] = 0
X = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
y=X%*%beta_true + rnorm(sample_size,mean=0,sd=2)
# Estimation
fit = GAGAs(X,y,alpha = 3,family="gaussian")
Eb = fit$beta
#Create testing data
X_t = R1*matrix(rnorm(sample_size * p_size), ncol = p_size)
y_t=X_t%*%beta_true + rnorm(sample_size,mean=0,sd=2)
#Prediction
Ey = predict.GAGA(fit,newx=X_t)

cat("\n err:", norm(Eb-beta_true,type="2")/norm(beta_true,type="2"))
cat("\n acc:", cal.w.acc(as.character(Eb!=0),as.character(beta_true!=0)))
cat("\n perr:", norm(Ey-y_t,type="2")/sqrt(sample_size))
```

`predict_cox_GAGA` *Get predictions from a GAGA cox model fit object*

Description

Get predictions from a GAGA cox model fit object

Usage

```
predict_cox_GAGA(fit, newx)
```

Arguments

`fit`	Fitted "GAGA" object.
`newx`	Matrix of new values for x at which predictions are to be made. Must be a matrix. If the intercept term needs to be considered in the estimation process, then the first column of `X` must be all 1s.

Value

Predictions

predict_LM_GAGA *Get predictions from a GAGA linear model fit object*

Description

Get predictions from a GAGA linear model fit object

Usage

```
predict_LM_GAGA(fit, newx)
```

Arguments

`fit`	Fitted "GAGA" object.
`newx`	Matrix of new values for x at which predictions are to be made. Must be a matrix. If the intercept term needs to be considered in the estimation process, then the first column of `X` must be all 1s.

Value

Predictions

predict_logistic_GAGA *Get predictions from a GAGA logistic model fit object*

Description

Get predictions from a GAGA logistic model fit object

Usage

```
predict_logistic_GAGA(fit, newx)
```

Arguments

`fit`	Fitted "GAGA" object.
`newx`	Matrix of new values for x at which predictions are to be made. Must be a matrix. If the intercept term needs to be considered in the estimation process, then the first column of `X` must be all 1s.

Value

Predictions

predict_multinomial_GAGA — *Get predictions from a GAGA multinomial model fit object*

Description

Get predictions from a GAGA multinomial model fit object

Usage

```
predict_multinomial_GAGA(fit, newx)
```

Arguments

`fit`	Fitted "GAGA" object.
`newx`	Matrix of new values for `x` at which predictions are to be made. Must be a matrix. If the intercept term needs to be considered in the estimation process, then the first column of `X` must be all 1s.

Value

Predictions

predict_poisson_GAGA — *Get predictions from a GAGA poisson model fit object*

Description

Get predictions from a GAGA poisson model fit object

Usage

```
predict_poisson_GAGA(fit, newx)
```

Arguments

`fit`	Fitted "GAGA" object.
`newx`	Matrix of new values for `x` at which predictions are to be made. Must be a matrix. If the intercept term needs to be considered in the estimation process, then the first column of `X` must be all 1s.

Value

Predictions

`rcpp_lm_gaga` *Fit a linear model via the GAGA algorithm using cpp.*

Description

Fit a linear model via the GAGA algorithm using cpp.

Usage

```
rcpp_lm_gaga(
  X,
  y,
  s_alpha,
  s_itrNum,
  s_thresh,
  s_QR_flag,
  s_flag,
  s_lamda_0,
  s_fix_sigma,
  s_sigm2_0,
  s_fdiag,
  s_frp
)
```

Arguments

`X`	Input matrix, of dimension nobs*nvars; each row is an observation. If the intercept term needs to be considered in the estimation process, then the first column of `X` must be all 1s.
`y`	Quantitative response N*1 matrix.
`s_alpha`	Hyperparameter. The suggested value for alpha is 2 or 3.
`s_itrNum`	The number of iteration steps. In general, 20 steps are enough.
`s_thresh`	Convergence threshold for beta Change, if `max(abs(beta-beta_old))<threshold`, return.
`s_QR_flag`	It identifies whether to use QR decomposition to speed up the algorithm.
`s_flag`	It identifies whether to make model selection. The default is `TRUE`.
`s_lamda_0`	The initial value of the regularization parameter for ridge regression.
`s_fix_sigma`	It identifies whether to update the variance estimate of the Gaussian noise or not.
`s_sigm2_0`	The initial variance of the Gaussian noise.
`s_fdiag`	It identifies whether to use diag Approximation to speed up the algorithm.
`s_frp`	Pre-processing by OMP method to reduce the number of parameters

Value

Coefficient vector

Index