Package ‘longfused’

January 29, 2015

Type Package
Title The Multinomial Fused Lasso Model
Version 1.0
Date 2014-10-15
Author Samrachana Adhikari, Fabrizio Lecci, Ryan Tibshirani
Maintainer Fabrizio Lecci <lecci@cmu.edu>
Description This package provides an efficient proximal gradient descent algorithm for computing solutions of the fused lasso regularized multinomial regression problem.
License GPL-3

R topics documented:

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>generatedata</td>
<td>Generate Data from a Multinomial Logit Model</td>
</tr>
</tbody>
</table>

Description
generatedata generates the outcome variable of a longitudinal multinomial logit model, starting from an array of coefficients beta and the matrix of (longitudinal) predictors X. The model is

\[
\log \frac{P(Y_{it} = 1|X_{i,t} = x)}{P(Y_{it} = K|X_{i,t} = x)} = \beta_{01} + \beta_{11}^T x
\]

\[
\log \frac{P(Y_{it} = 2|X_{i,t} = x)}{P(Y_{it} = K|X_{i,t} = x)} = \beta_{02} + \beta_{12}^T x
\]

\[
\vdots
\]

\[
\log \frac{P(Y_{it} = K - 1|X_{i,t} = x)}{P(Y_{it} = K|X_{i,t} = x)} = \beta_{0(K-1)} + \beta_{1(K-1)}^T x
\]
Usage

generateData(X, beta)

Arguments

X an n by p by TT array of predictors, where n is the number of individuals and p is the number of predictors observed at TT different times.

beta a (p+1) by TT by (K-1) array coefficients, where K is the number of levels of the outcome variable. The matrix beta[1, , ] accommodates the initial the intercept values.

Details

See the reference.

Value

returns an n by TT matrix of outcomes, whose possible values are 1, . . . , K.

Author(s)

Fabrizio Lecci

References


Examples

n=50  ## number of individuals
p=30  ## number of predictors
TT=15  ## number of time points
KK=2  ## levels of the outcome variable

###################################################
## Generating TRUE beta
###################################################
beta[2,] = matrix(rep(c(5,5,5,5,5,3,3,3,3,3,0,0,0,0),KK-1), ncol=(KK-1))
beta[3,] = matrix(rep(c(0,0,0,0,-4,-4,-4,-4,-4,-4,-4,-4),KK-1), ncol=(KK-1))
beta[4,] = matrix(rep(c(-3,-3,-3,-3,-3,4,4,4,4,4,4,4,4,4,4,4,4),KK-1), ncol=(KK-1))

###################################################
## Generate some data
###################################################
X = array(rnorm(n*(p+TT,0,1)),dim=c(n,(p),TT))
Y = generateData(X,beta)
Description

`longfused` estimates the coefficients of a multinomial logit model with lasso and fused lasso penalties, using a Generalized Gradient Descent algorithm.

Usage

`longfused(X, Y, betaInit, lambda1, lambda2, niter, stop, eps, tauStart, factor, scaleLoss=FALSE)`

Arguments

- **X**: an n by p by T array of predictors, where n is the number of individuals and p is the number of predictors observed at T different times.
- **Y**: an n by T matrix of outcomes, whose possible values are 1, ..., K, where K is the number of categories.
- **betaInit**: a (p+1) by T by (K-1) array of initial coefficient guesses. The matrix `betaInit[1, ,]` accommodates the initial guesses for the intercepts.
- **lambda1**: number: the lasso penalty.
- **lambda2**: number: the fused lasso penalty.
- **niter**: maximum number of iterations in the Generalized Gradient Descent Algorithm.
- **stop**: if 1: stopping criterion on the objective function is used. if 2: stopping criterion on the betas is used.
- **eps**: tolerance level for the stopping criterion.
- **tauStart**: initial step size before backtracking.
- **factor**: backtracking shrinkage parameter.
- **scaleLoss**: if TRUE the loss function is computed only on observed individuals and each term of the loss is divided by the number of completely observed individuals at time t. This fix makes the loss (roughly) independent of the effective sample size.

Details

See the reference.

Value

Returns an object of class "longfused", a list with the following elements:

- **betaHat**: a (p+1) by T by (K-1) array of estimated coefficients.
- **iter**: number of iterations of the GGD algorithm.
- **objective**: values of the objective function to be minimized, for each step of the GGD algorithm.
- **tau**: step size for each step of the GGD algorithm.
- **StopCriterion**: criterion’s value for the stopping criterion, for each step of the GGD algorithm.
Author(s)
Fabrizio Lecci, Samrachana Adhikari, Ryan Tibshirani

References

Examples
n
p

r

q

Examples

# Results
rslt1 <- longfused(X, Y, beta = init, lambda1 = 0, lambda2 = 0, niter = 50, stop = 1, eps = 0.000001, taustart = 0.000001, factor = 0.8)

# longfused estimation
# lambda1 very small (almost equivalent to simple logistic regression)
# lambda2 = 0

# Generating TRUE beta
beta = array(0, c(p, 1, TT, (KK - 1)))
betainit = matrix(0, c(p, TT, (KK - 1)))

# Generate some data
X = matrix(rnorm(n * p * tt), ncol = tt)
Y = generateData(X, beta)

# Longitudinal estimation with positive lambda1 and lambda2
niter = 50
betainit = matrix(0, c(p, TT, (KK - 1)))
lambda1 = P6
lambda2 = PRU
factor = PNX
eps = 0.000001

stop = 1
scaleloss = FALSE

# Results
rslt2 <- longfused(X, Y, beta = init, lambda1 = 0, lambda2 = 0, niter = 50, stop = 1, eps = 0.000001, taustart = 0.000001, factor = 0.8)
**plot.longfused**

```r
eps=0.000001
tauStart=260
stop=1

## Results
rslt2<-longfused(X,Y,betaInit,lambda1,lambda2,niter,stop,eps,tauStart,factor, scaleLoss=FALSE)

par(mfrow=c(1,3), mar=c(4.5,4.5,2,0))
trueBeta=list(betaHat=beta)
class(trueBeta)="longfused"
plot(trueBeta, xlab="Time", ylab="True beta")
plot(rs1t1, xlab="Time", ylab="Estimated beta")
plot(rs1t2, xlab="Time", ylab="Estimated beta")
```

---

**plot.longfused**  
*Plot longfused estimated coefficients.*

**Description**

This function plots the estimated coefficients stored in an object of class `longfused`.

**Usage**

```r
## S3 method for class 'longfused'
plot(x, intercept=FALSE, levels=NULL, beta=NULL, color=NULL, ltype=NULL, ...)```

**Arguments**

- `x` an object of class "longfused", as returned by the function `longfused`.
- `intercept` if `TRUE` the intercept is displayed.
- `levels` a vector of strings for the levels of the outcome variable in the longfused model.
- `beta` Optional: a `(p+1)` by `TT` by `(K-1)` array of true coefficients to be displayed along with the estimated coefficients. Default is `NULL` and the true coefficients are not displayed.
- `color` vector of length `p+1` storing the colors of the coefficients trajectories
- `ltype` vector of length `p+1` storing the line type of the coefficients trajectories
- `...` additional graphical parameters.

**Author(s)**

Fabrizio Lecci
Examples

```r
n=50  ## number of individuals
p=30  ## number of predictors
TT=15  ## number of time points
KK=2  ## levels of the outcome variable

# Generating TRUE beta
beta=matrix(rep(c(5,5,5,5,3,3,3,3,0,0,0,0,0,0,0,0),KK-1), ncol=(KK-1))
beta[2,]=matrix(rep(c(0,0,0,0,-4,-4,-4,-4,-4,-4,-4,-4),KK-1), ncol=(KK-1))
beta[4,]=matrix(rep(c(-3,-3,-3,-3,3,4,4,4,4,4,4,4),KK-1), ncol=(KK-1))

# Generate some data
X=array(rnorm(n*p*TT,0,1),dim=c(n,(p),TT))
Y=generateData(X,beta)

# longfused estimation
# lambda1 very small (almost equivalent to simple logistic regression)
# lambda2=0
niter=500
betainit= array(0,c(p+1,TT,(KK-1)))
lambda1=0
lambda2=0
factor = 0.8
eps=0.00000001
tauStart=260
stop=1

# Results
rslt1<- longfused(X,Y,betainit,lambda1,lambda2,niter,stop,eps,tauStart,factor, scaleLoss=FALSE)

# ESTIMATION TEST with positive lambda1 and lambda2
niter=500
betainit= array(0,c(p+1,TT,(KK-1)))
lambda1=0.06+50
lambda2=0.25+50
factor = 0.8
eps=0.000001
tauStart=260
stop=1

# Results
rslt2<- longfused(X,Y,betainit,lambda1,lambda2,niter,stop,eps,tauStart,factor, scaleLoss=FALSE)
```

# Plot true coefficients and 2 different results
```r
par(mfrow=c(1,3), mar=c(4.5,4.5,2,0))
trueBeta=list(betahat=beta)
class(trueBeta)="longfused"
plot(trueBeta, xlab="Time", ylab="True beta")
plot(rslt1, xlab="Time", ylab="Estimated beta")
plot(rslt2, xlab="Time", ylab="Estimated beta")
```
Index

generateData, 1
longfused, 3, 5
plot.longfused, 5