Package 'APML0'

April 15, 2018

Type Package

Title Augmented and Penalized Minimization Method L0

Version 0.8

Author Xiang Li, Shanghong Xie, Donglin Zeng and Yuanjia Wang

Maintainer Xiang Li <xli256@its.jnj.com>

Description Fit linear, logistic and Cox models regularized with L0, lasso (L1), elastic-net (L1 and L2), or net (L1 and Laplacian) penalty, and their adaptive forms, such as adaptive lasso / elastic-net and net adjusting for signs of linked coefficients. It solves L0 penalty problem by simultaneously selecting regularization parameters and the number of non-zero coefficients. This augmented and penalized minimization method provides an approximation solution to the L0 penalty problem, but runs as fast as L1 regularization problem. The package uses one-step coordinate descent algorithm and runs extremely fast by taking into account the sparsity structure of coefficients. It could deal with very high dimensional data and has superior selection performance.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports Rcpp (>= 0.12.12)

LinkingTo Rcpp, RcppEigen

Depends Matrix (>= 1.2-10)

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-04-15 19:20:33 UTC

R topics documented:

APML0-package *Augmented and Penalized Minimization Method L0*

Description

Fit linear, logistic and Cox models regularized with L0, lasso (L1), elastic-net (L1 and L2), or net (L1 and Laplacian) penalty, and their adaptive forms, such as adaptive lasso / elastic-net and net adjusting for signs of linked coefficients. It solves L0 penalty problem by simultaneously selecting regularization parameters and the number of non-zero coefficients. This augmented and penalized minimization method provides an approximation solution to the L0 penalty problem, but runs as fast as L1 regularization problem.

The package uses one-step coordinate descent algorithm and runs extremely fast by taking into account the sparsity structure of coefficients. It could deal with very high dimensional data.

Details

Package:	APML0
Type:	Package
Version:	0.8
Date:	2018-04-01
License:	GPL (>= 2)

Functions: APML0, print.APML0

Author(s)

Xiang Li, Shanghong Xie, Donglin Zeng and Yuanjia Wang
Maintainer: Xiang Li <xli256@its.jnj.com>

References

Li, X., Xie, S., Zeng, D., Wang, Y. (2018). *Efficient l0-norm feature selection based on augmented and penalized minimization. Statistics in Medicine, 37(3), 473-486.*
Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). *Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning, 3(1), 1-122.*
http://dl.acm.org/citation.cfm?id=2185816
Friedman, J., Hastie, T. and Tibshirani, R. (2010). *Regularization paths for generalized linear models via coordinate descent, Journal of Statistical Software, Vol. 33(1), 1.*
http://www.jstatsoft.org/v33/i01/

Examples

```
###  Linear model  ###
set.seed(1213)
N=100;p=30;p1=5
x=matrix(rnorm(N*p),N,p)
beta=rnorm(p1)
xb=x[,1:p1]%*%beta
y=rnorm(N,xb)

fiti=APML0(x,y,penalty="Lasso",nlambda=10) # Lasso
fiti2=APML0(x,y,penalty="Lasso",nlambda=10,nfolds=10) # Lasso
# attributes(fiti)


###  Logistic model  ###
set.seed(1213)
N=100;p=30;p1=5
x=matrix(rnorm(N*p),N,p)
beta=rnorm(p1)
xb=x[,1:p1]%*%beta
y=rbinom(n=N, size=1, prob=1.0/(1.0+exp(-xb)))

fiti=APML0(x,y,family="binomial",penalty="Lasso",nlambda=10) # Lasso
fiti2=APML0(x,y,family="binomial",penalty="Lasso",nlambda=10,nfolds=10) # Lasso
# attributes(fiti)


###  Cox model  ###
set.seed(1213)
N=100;p=30;p1=5
x=matrix(rnorm(N*p),N,p)
beta=rnorm(p1)
xb=x[,1:p1]%*%beta
ty=rexp(N,exp(xb))
tcens=rbinom(n=N,prob=.3,size=1)  # censoring indicator
y=cbind(time=ty,status=1-tcens)

fiti=APML0(x,y,family="cox",penalty="Lasso",nlambda=10) # Lasso
fiti2=APML0(x,y,family="cox",penalty="Lasso",nlambda=10,nfolds=10) # Lasso
# attributes(fiti)
```

APML0 *Fit a Model with Various Regularization Forms*

Description

Fit linear, logistic and Cox models regularized with L0, lasso (L1), elastic-net (L1 and L2), or net (L1 and Laplacian) penalty, and their adaptive forms, such as adaptive lasso / elastic-net and net adjusting for signs of linked coefficients. It solves L0 penalty problem by simultaneously selecting

regularization parameters and the number of non-zero coefficients. This augmented and penalized minimization method provides an approximation solution to the L0 penalty problem, but runs as fast as L1 regularization problem.

The package uses one-step coordinate descent algorithm and runs extremely fast by taking into account the sparsity structure of coefficients. It could deal with very high dimensional data.

Usage

```
APML0(x, y, family=c("gaussian", "binomial", "cox"), penalty=c("Lasso","Enet", "Net"),
Omega=NULL, alpha=1.0, lambda=NULL, nlambda=50, rlambda=NULL,
wbeta=rep(1,ncol(x)), sgn=rep(1,ncol(x)), nfolds=1, foldid=NULL, inzero=TRUE,
isd=FALSE, iysd=FALSE, keep.beta=FALSE, ifast=TRUE, thresh=1e-7, maxit=1e+5)
```

Arguments

x	input matrix. Each row is an observation vector.
y	response variable. For `family = "gaussian"`, y is a continuous vector. For `family = "binomial"`, y is a binary vector with 0 and 1. For `family = "cox"`, y is a two-column matrix with columns named 'time' and 'status'. 'status' is a binary variable, with '1' indicating event, and '0' indicating right censored.
family	type of outcome. Can be "gaussian", "binomial" or "cox".
penalty	penalty type. Can choose `"Net"`, `"Enet"` (elastic net) and `"Lasso"`. For `"Net"`, need to specify Omega; otherwise, `"Enet"` is performed. For `penalty = "Net"`, the penalty is defined as $$\lambda * \alpha * ||\beta||_1 + (1-\alpha)/2 * (\beta^T L \beta),$$ where L is a Laplacian matrix calculated from Omega.
Omega	adjacency matrix with zero diagonal and non-negative off-diagonal, used for `penalty = "Net"` to calculate Laplacian matrix.
alpha	ratio between L1 and Laplacian for `"Net"`, or between L1 and L2 for `"Enet"`. Default is `alpha = 1.0`, i.e. lasso.
lambda	a user supplied decreasing sequence. If `lambda = NULL`, a sequence of lambda is generated based on nlambda and rlambda. Supplying a value of lambda overrides this.
nlambda	number of lambda values. Default is 50.
rlambda	fraction of `lambda.max` to determine the smallest value for lambda. The default is `rlambda = 0.0001` when the number of observations is larger than or equal to the number of variables; otherwise, `rlambda = 0.01`.
wbeta	penalty weights used with L1 penalty (adaptive L1), given by $\sum_{j=1}^{q} w_j \|\beta_j\|$. The wbeta is a vector of non-negative values and works as adaptive L1. No penalty is imposed for those coefficients with zero values in wbeta. Default is 1 for all coefficients.
sgn	sign adjustment used with Laplacian penalty (adaptive Laplacian). The sgn is a vector of 1 or -1. The sgn could be based on an initial estimate of β, and 1 is used for $\beta > 0$ and -1 is for $\beta < 0$. Default is 1 for all coefficients.

nfolds	number of folds. With nfolds = 1 and foldid = NULL by default, cross-validation is not performed. For cross-validation, smallest value allowable is nfolds = 3. Specifying foldid overrides nfolds.
foldid	an optional vector of values between 1 and nfolds specifying which fold each observation is in.
inzero	logical flag for simultaneously selecting the number of non-zero coefficients with lambda. Default is inzero = TRUE.
isd	logical flag for outputting standardized coefficients. x is always standardized prior to fitting the model. Default is isd = FALSE, returning β on the original scale.
iysd	logical flag for standardizing y prior to computation, for family = "gaussian". The returning coefficients are always based the original y (unstandardized). Default is isd = FALSE.
keep.beta	logical flag for returning estimates for all lambda values. For keep.beta = FALSE, only return the estimate with the minimum cross-validation value.
ifast	logical flag for efficient calculation of risk set updates for family = "cox". Default is ifast = TRUE.
thresh	convergence threshold for coordinate descent. Default value is 1E-7.
maxit	Maximum number of iterations for coordinate descent. Default is 10^5.

Details

One-step coordinate descent algorithm is applied for each lambda. For family = "cox", ifast = TRUE adopts an efficient way to update risk set and sometimes the algorithm ends before all nlambda values of lambda have been evaluated. To evaluate small values of lambda, use ifast = FALSE. The two methods only affect the efficiency of algorithm, not the estimates.

x is always standardized prior to fitting the model and the estimate is returned on the original scale. For family = "gaussian", y is centered by removing its mean, so there is no intercept output.

Cross-validation is used for tuning parameters. For inzero = TRUE, we further select the number of non-zero coefficients obtained from regularized model at each lambda. This is motivated by formulating L0 variable selection in an augmented form, which shows significant improvement over the commonly used regularized methods without this technique.

Value

An object with S3 class "APML0".

Beta	a sparse Matrix of coefficients, stored in class "dgCMatrix". For family = "binomial", the first coefficient is the intercept.
Beta0	coefficients after additionally tuning the number of non-zeros, for inzero = TRUE. For family = "binomial", the first coefficient is the intercept.
fit	a data.frame containing lambda and the number of non-zero coefficients nzero. With cross-validation, additional results are reported, such as average cross-validation partial likelihood cvm and its standard error cvse, and index with '*' indicating the minimum cvm. For family = "gaussian", rsq is also reported.

fit0	a data.frame containing lambda, cvm and nzero based on inzero = TRUE. cvm in fit0 may be different from cvm in fit, because the constaint on the number of non-zeros is imposed in the cross-validation. The maximum number of non-zeros is based on the full dataset not the one used in the cross-validation.
lambda.min	value of lambda that gives minimum cvm.
lambda.opt	value of lambda based on inzero = TRUE.
penalty	penalty type.
adaptive	logical flags for adaptive version (see above).
flag	convergence flag (for internal debugging). flag = 0 means converged.

Warning

It may terminate and return NULL.

Author(s)

Xiang Li, Shanghong Xie, Donglin Zeng and Yuanjia Wang
Maintainer: Xiang Li <xli256@its.jnj.com>, Shanghong Xie <sx2168@cumc.columbia.edu>

References

Li, X., Xie, S., Zeng, D., Wang, Y. (2017). *Efficient Method to Optimally Identify Important Biomarkers for Disease Outcomes with High-dimensional Data. Statistics in Medicine, accepted.*
Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). *Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning, 3(1), 1-122.*
http://dl.acm.org/citation.cfm?id=2185816
Friedman, J., Hastie, T. and Tibshirani, R. (2010). *Regularization paths for generalized linear models via coordinate descent, Journal of Statistical Software, Vol. 33(1), 1.*
http://www.jstatsoft.org/v33/i01/

See Also

print.APML0

Examples

```
###  Linear model  ###
set.seed(1213)
N=100;p=30;p1=5
x=matrix(rnorm(N*p),N,p)
beta=rnorm(p1)
xb=x[,1:p1]%*%beta
y=rnorm(N,xb)

fiti=APML0(x,y,penalty="Lasso",nlambda=10) # Lasso
fiti2=APML0(x,y,penalty="Lasso",nlambda=10,nfolds=10) # Lasso
# attributes(fiti)
```

```
###  Logistic model  ###
set.seed(1213)
N=100;p=30;p1=5
x=matrix(rnorm(N*p),N,p)
beta=rnorm(p1)
xb=x[,1:p1]%*%beta
y=rbinom(n=N, size=1, prob=1.0/(1.0+exp(-xb)))

fiti=APML0(x,y,family="binomial",penalty="Lasso",nlambda=10) # Lasso
fiti2=APML0(x,y,family="binomial",penalty="Lasso",nlambda=10,nfolds=10) # Lasso
# attributes(fiti)



###  Cox model  ###
set.seed(1213)
N=100;p=30;p1=5
x=matrix(rnorm(N*p),N,p)
beta=rnorm(p1)
xb=x[,1:p1]%*%beta
ty=rexp(N,exp(xb))
tcens=rbinom(n=N,prob=.3,size=1)  # censoring indicator
y=cbind(time=ty,status=1-tcens)

fiti=APML0(x,y,family="cox",penalty="Lasso",nlambda=10) # Lasso
fiti2=APML0(x,y,family="cox",penalty="Lasso",nlambda=10,nfolds=10) # Lasso
# attributes(fiti)
```

`print.APML0` *Print a APML0 Object*

Description

Print a summary of results along the path of `lambda`.

Usage

```
## S3 method for class 'APML0'
print(x, digits = 4, ...)
```

Arguments

`x`	fitted APML0 object
`digits`	significant digits in printout
`...`	additional print arguments

Details

The performed model is printed, followed by `fit` and `fit0` (if any) from a fitted `APML0` object.

Value

The data frame above is silently returned

Author(s)

Xiang Li, Shanghong Xie, Donglin Zeng and Yuanjia Wang
Maintainer: Xiang Li <xli256@its.jnj.com>

See Also

`APML0`

Examples

```
###  Linear model  ###
set.seed(1213)
N=100;p=30;p1=5
x=matrix(rnorm(N*p),N,p)
beta=rnorm(p1)
xb=x[,1:p1]%*%beta
y=rnorm(N,xb)

fiti2=APML0(x,y,penalty="Lasso",nlambda=10,nfolds=10) # Lasso
fiti2
```

Index