Package 'ADMMnet'

December 12, 2015

Type Package

Title Regularized Model with Selecting the Number of Non-Zeros

Version 0.1

Date 2015-12-10

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

Maintainer Xiang Li <xl2473@cumc.columbia.edu>

Description Fit linear and cox models regularized with net (L1 and Laplacian), elastic-net (L1 and L2) or lasso (L1) penalty, and their adaptive forms, such as adaptive lasso and net adjusting for signs of linked coefficients. In addition, it treats the number of non-zero coefficients as another tuning parameter and simultaneously selects with the regularization parameter. The package uses one-step coordinate descent algorithm and runs extremely fast by taking into account the sparsity structure of coefficients.

License GPL (>= 2)

Imports Rcpp (>= 0.12.2)

LinkingTo Rcpp, RcppEigen

Depends Matrix (>= 1.2-3)

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-12-12 09:41:54

R topics documented:

`ADMMnet-package` *Regularized Model with Selecting the Number of Non-Zeros*

Description

This package fits linear and cox models regularized with net (L1 and Laplacian), elastic-net (L1 and L2) or lasso (L1) penalty, and their adaptive forms, such as adaptive lasso and net adjusting for signs of linked coefficients. In addition, it treats the number of non-zero coefficients as another tuning parameter and simultaneously selects with the regularization parameter `lambda`. This is motivated by formulating L0 variable selection in ADMM form. By selecting the regularization parameter and the number of non-zeros, it shows significant improvement over the commonly used regularized methods, which depend on the regularization parameter only.

The package uses one-step coordinate descent algorithm and runs extremely fast by taking into account the sparsity structure of coefficients.

Details

Package:	ADMMnet
Type:	Package
Version:	0.1
Date:	2015-12-10
License:	GPL (>= 2)

Functions: `ADMMnet`, `print.ADMMnet`

Author(s)

Xiang Li, Shanghong Xie, Donglin Zeng and Yuanjia Wang
Maintainer: Xiang Li <xl2473@cumc.columbia.edu>

References

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/`
Li, C., and Li, H. (2010). *Variable selection and regression analysis for graph-structured covariates with an application to genomics. The annals of applied statistics, 4(3), 1498.*
`http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3423227/`
Sun, H., Lin, W., Feng, R., and Li, H. (2014) *Network-regularized high-dimensional cox regression for analysis of genomic data, Statistica Sinica.*

http://www3.stat.sinica.edu.tw/statistica/j24n3/j24n319/j24n319.html

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]
y=rnorm(N,xb)

fiti=ADMMnet(x,y,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]
ty=rexp(N,exp(xb))
tcens=rbinom(n=N,prob=.3,size=1)  # censoring indicator
y=cbind(time=ty,status=1-tcens)

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

ADMMnet *Fit a Model with Various Regularization Forms*

Description

Fit a linear or cox model regularized with net (L1 and Laplacian), elastic-net (L1 and L2) or lasso (L1) penalty, and their adaptive forms, such as adaptive lasso and net adjusting for signs of linked coefficients. In addition, it treats the number of non-zero coefficients as another tuning parameter and simultaneously selects with the regularization parameter `lambda`. The package uses one-step coordinate descent algorithm and runs extremely fast by taking into account the sparsity structure of coefficients.

Usage

```
ADMMnet(x, y, family = c("gaussian", "cox"), penalty = c("Lasso", "Enet", "Net"),
  Omega = NULL, alpha = 1.0, lambda = NULL, nlambda = 50, rlambda = NULL,
  nfolds = 1, foldid = NULL, inzero = TRUE, adaptive = c(FALSE, TRUE), aini = NULL,
  isd = FALSE, keep.beta = FALSE, ifast = TRUE, thresh = 1e-07, maxit = 1e+05)
```

Arguments

`x`	input matrix. Each row is an observation vector.
`y`	response variable. For `family = "gaussian"`, `y` is a continuous vector. 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" or "cox".
`penalty`	penalty type. Can choose `"Net"`, `"Enet"` (elastic net) and `"Lasso"`. For `"Net"`, need to specify `Omega`; otherwises, `"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`	correlation/adjacency matrix with zero 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 sequency 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`.
`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` overrisdes `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`.
`adaptive`	logical flags for adaptive version. Default is `adaptive = c(FALSE, TRUE)`. The first element is for adaptive on β in L1 and the second for adjusting for signs of linked coefficients in Laplacian matrix.
`aini`	a user supplied initial estimate of β. It is a list including `wbeta` for adaptive L1 and `sgn` for adjusting Laplacian matrix. `wbeta` is the absolute value of inverse initial estimates. If `aini = NULL` but `adaptive` is required, `aini` is generated from regularized model with `penatly = "Enet"` and `alpha = 0.0`, i.e. a ridge regression.
`isd`	logical flag for outputing standardized coefficients. `x` is always standardized prior to fitting the model. Default is `isd = FALSE`, returning β on the original scale.

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 ADMM form, which shows significant improvement over the commonly used regularized methods without this technique.

Value

An object with S3 class "ADMMnet".

Beta	a sparse Matrix of coefficients, stored in class "dgCMatrix".
Beta0	coefficients after additionally tuning the number of non-zeros, for inzero = TRUE.
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.
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 <xl2473@cumc.columbia.edu>, Shanghong Xie <sx2168@cumc.columbia.edu>

References

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/
Li, C., and Li, H. (2010). *Variable selection and regression analysis for graph-structured covariates with an application to genomics. The annals of applied statistics, 4(3), 1498.*
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3423227/
Sun, H., Lin, W., Feng, R., and Li, H. (2014) *Network-regularized high-dimensional cox regression for analysis of genomic data, Statistica Sinica.*
http://www3.stat.sinica.edu.tw/statistica/j24n3/j24n319/j24n319.html

See Also

print.ADMMnet

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]
y=rnorm(N,xb)

fiti=ADMMnet(x,y,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]
ty=rexp(N,exp(xb))
tcens=rbinom(n=N,prob=.3,size=1)  # censoring indicator
y=cbind(time=ty,status=1-tcens)

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

`print.ADMMnet` *Print a ADMMnet Object*

Description

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

Usage

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

Arguments

`x`	fitted ADMMnet 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 `ADMMnet` object.

Value

The data frame above is silently returned

Author(s)

Xiang Li, Shanghong Xie, Donglin Zeng and Yuanjia Wang
Maintainer: Xiang Li <xl2473@cumc.columbia.edu>

See Also

`ADMMnet`

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]
y=rnorm(N,xb)

fiti=ADMMnet(x,y,penalty="Lasso",nlambda=10,nfolds=10) # Lasso
fiti
```

Index