Package: APML0 (via r-universe)

June 20, 2026

Type Package

Title Augmented and Penalized Minimization Method L0

Version 0.11

Date 2026-06-13

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 the L0 penalty problem by simultaneously selecting regularization parameters and performing hard-thresholding or selecting the number of non-zeros. This augmented and penalized minimization method provides an approximation solution to the L0 penalty problem, but runs as fast as L1 regularization. The package uses a one-step coordinate descent algorithm and runs extremely fast by taking into account the sparsity structure of coefficients. It can handle very high dimensional data and has superior selection performance.

License GPL (>= 2)

Encoding UTF-8

URL https://github.com/LeeSprite/APML0

BugReports https://github.com/LeeSprite/APML0/issues

Imports Rcpp (>= 0.12.12)

LinkingTo Rcpp, RcppEigen

Depends Matrix (>= 1.2-10)

Config/roxygen2/version 8.0.0

Suggests spelling

Language en-US

Repository https://leesprite.r-universe.dev

Date/Publication 2026-06-13 15:27:30 UTC

RemoteUrl https://github.com/leesprite/apml0

RemoteRef HEAD

RemoteSha e1495a087a6a61d4f141b2581ceb03e7f456a122

Contents

`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. The package solves the L0 penalty problem by simultaneously selecting regularization parameters and performing hard-thresholding or selecting the number of non-zeros. This augmented and penalized minimization method provides an approximation solution to the L0 penalty problem but runs as fast as L1 regularization. A one-step coordinate descent algorithm exploits sparsity and handles very high dimensional data efficiently.

Details

Package:	APML0
Type:	Package
Version:	0.11
Date:	2026-06-13
License:	GPL (>= 2)

Main function: `APML0`
Print method: `print.APML0`

Author(s)

Xiang Li, Shanghong Xie, Donglin Zeng and Yuanjia Wang
Maintainer: Xiang Li <spiritcoke@gmail.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. doi:10.1002/sim.7526

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.

Friedman, J., Hastie, T., Tibshirani, R. (2010). *Regularization paths for generalized linear models via coordinate descent.* Journal of Statistical Software, 33(1). doi:10.18637/jss.v033.i01

Examples

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

`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 the L0 penalty problem by simultaneously selecting regularization parameters and performing hard-thresholding (or selecting the number of non-zeros). This augmented and penalized minimization method provides an approximation solution to the L0 penalty problem and runs as fast as L1 regularization.

The function uses a one-step coordinate descent algorithm and runs extremely fast by taking into account the sparsity structure of coefficients. It can handle very high dimensional data.

Usage

```
APML0(x, y, weights=NULL, 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)), KK=log(nrow(x)),
nfolds=1, foldid=NULL, cvll=TRUE, icutB=TRUE, ncutB=10,
ifast=TRUE, isd=FALSE, iysd=FALSE, ifastr=TRUE, keep.beta=FALSE,
thresh=1e-6, maxit=1e+5, threshC=1e-5, maxitC=1e+2, threshP=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 binary, with 1 indicating event and 0 indicating right censoring.
weights	case weights for each observation. Default is 1 for every observation. For family = "cox", observations with weight zero are excluded from the risk set.
family	type of outcome. One of "gaussian", "binomial", or "cox".
penalty	penalty type. One of "Lasso", "Enet" (elastic net), or "Net". For "Net", Omega must be supplied; otherwise "Enet" is used. For penalty = "Net", the penalty is $$\lambda\{\alpha\|\beta\|_1 + (1-\alpha)/2\,\beta^T L\beta\},$$ where L is the Laplacian matrix computed from Omega.
Omega	adjacency matrix with zero diagonal and non-negative off-diagonal, used with penalty = "Net".
alpha	mixing parameter between L1 and Laplacian for "Net", or between L1 and L2 for "Enet". Default alpha = 1.0 (lasso).
lambda	user-supplied decreasing sequence of regularization values. If NULL, a sequence is generated from nlambda and rlambda.
nlambda	number of lambda values. Default is 50.
rlambda	fraction of lambda.max used as the smallest lambda. Default is 0.0001 when $n \geq p$ and 0.01 otherwise.
wbeta	penalty weights for the L1 term (adaptive L1): $\sum_j w_j\|\beta_j\|$. Zero entries remove the penalty on the corresponding coefficient. The same weights are applied to the L0 step. Default is 1 for all coefficients.
sgn	sign adjustment for the Laplacian penalty (adaptive Laplacian). A vector of 1 or -1. Default is 1 for all coefficients.
KK	multiplier in the BIC-type criterion used when no cross-validation is performed (nfolds = 1 and foldid = NULL). The criterion is $nzero \times KK - 2 \times \mathrm{logLik}$. Default KK = log(nrow(x)) gives BIC; KK = 2 gives AIC.
nfolds	number of cross-validation folds. With nfolds = 1 and foldid = NULL (default), BIC-type selection is used. Minimum value for cross-validation is 3. Specifying foldid overrides nfolds.
foldid	optional vector of fold assignments (integers 1 to nfolds) for each observation.
cvll	logical flag for using the log-likelihood as the cross-validation criterion. Default TRUE. For family = "gaussian", set FALSE to use mean squared prediction error. Not used for family = "cox" (partial likelihood is always used).
icutB	logical flag for L0 selection via hard-thresholding (TRUE, default) rather than by selecting the number of non-zeros (FALSE). Applies to all three families.
ncutB	number of threshold grid points for icutB = TRUE. Default is 10. Larger values may improve selection but increase runtime.

isd	logical flag for returning standardized coefficients. x is always standardized internally; isd = FALSE (default) returns β on the original scale.
iysd	logical flag for standardizing y before fitting, for family = "gaussian" only. Returned coefficients are always on the original y scale. Default is FALSE.
keep.beta	logical flag for returning estimates at all lambda values. When FALSE (default and cross-validation only), only the estimate at the optimal lambda is returned.
ifast	logical flag for the fast computation path. When TRUE (default), BIC/CV selection reuses the regularized fit without re-fitting. Set FALSE to re-fit via maximum likelihood on the selected support.
ifastr	logical flag for family = "cox" only. When TRUE (default), an efficient risk-set update is used; the algorithm may stop before all nlambda values are evaluated. Affects only efficiency, not the estimates.
thresh	convergence threshold for coordinate descent. Default 1e-6.
maxit	maximum coordinate descent iterations. Default 1e5.
threshC	convergence threshold for the hard-thresholding step. Default 1e-5.
maxitC	maximum iterations for the hard-thresholding step. Default 100.
threshP	probability truncation bound for family = "binomial". Default 1e-5.

Details

A one-step coordinate descent algorithm is applied for each lambda. For family = "cox", ifastr = TRUE uses an efficient risk-set update and may stop before all nlambda values are evaluated; use ifastr = FALSE to force evaluation of all values.

x is always standardized internally and estimates are returned on the original scale. For family = "gaussian", y is centered (no intercept is returned).

L0 variable selection is always performed in addition to the regularized fit. Without cross-validation, the support is selected by a BIC-type criterion $nzero \times KK - 2 \times \text{logLik}$ (see KK). With cross-validation, selection uses hard-thresholding (icutB = TRUE) or the number of non-zeros (icutB = FALSE).

Value

An object of S3 class "APML0" containing:

Beta0	coefficients after L0 selection. For family = "binomial", the first element is the intercept.
Beta	sparse matrix of regularized coefficients (class "dgCMatrix") along the lambda path. For family = "binomial", the first row is the intercept.
fit	data frame with columns lambda and nzero. Without cross-validation: also criterion (BIC-type score). With cross-validation: also cvm, cvse, and index. For family = "gaussian": also rsq.
fit0	data frame for the L0-selected solution: lambda, cvm (or criterion), nzero.
lambda.min	lambda giving minimum cvm (cross-validation only).
lambda.opt	lambda for the L0 solution (cross-validation only).

penalty	penalty type used.
adaptive	logical flag(s) for adaptive weighting.
flag	convergence flag. 0 means converged.

Warning

The function may return NULL if no valid lambda values are found.

Author(s)

Xiang Li, Shanghong Xie, Donglin Zeng and Yuanjia Wang
Maintainer: Xiang Li <spiritcoke@gmail.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. doi:10.1002/sim.7526

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.

Friedman, J., Hastie, T., Tibshirani, R. (2010). *Regularization paths for generalized linear models via coordinate descent.* Journal of Statistical Software, 33(1). doi:10.18637/jss.v033.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)
fiti2=APML0(x, y, penalty="Lasso", nlambda=10, nfolds=10)


###  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)
```

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


###  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))
td=rexp(N, 0.05)
tcens=ifelse(td < ty, 1, 0)
y=cbind(time=ty, status=1-tcens)

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

`print.APML0` *Print an APML0 Object*

Description

Print a summary of results along the regularization path.

Usage

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

Arguments

`x`	a fitted `APML0` object.
`digits`	number of significant digits in the output. Default is 4.
`...`	additional arguments (currently unused).

Details

Prints the model family and penalty type, the `fit` table along the `lambda` path, and, when cross-validation with L0 selection was performed, the `fit0` table for the selected solution.

Value

Called for its side effect (printing). Returns `x` invisibly.

Author(s)

Xiang Li, Shanghong Xie, Donglin Zeng and Yuanjia Wang
Maintainer: Xiang Li <spiritcoke@gmail.com>

See Also

APML0

Examples

```
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, nfolds=10)
print(fiti)
```

Index