Package ‘PUlasso’

February 13, 2026

Type Package

Title High-Dimensional Variable Selection with Presence-Only Data

Version 3.2.6

Date 2026-2-11

Description Efficient algorithm for solving PU (Positive and Unlabeled) problem in low or high dimensional setting with lasso or group lasso penalty. The algorithm uses Maximization-Minorization and (block) coordinate descent. Sparse calculation and parallel computing are supported for the computational speed-up. See Hyebin Song, Garvesh Raskutti (2018) <doi:10.48550/arXiv.1711.08129>.

License GPL-2

Imports Rcpp (>= 0.12.8), methods, Matrix, doParallel, foreach, ggplot2

Depends R(>= 2.10)

LinkingTo Rcpp, RcppEigen, Matrix

RoxygenNote 7.3.2

Encoding UTF-8

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

URL `https://arxiv.org/abs/1711.08129`

BugReports `https://github.com/hsong1/PUlasso/issues`

NeedsCompilation yes

Author Hyebin Song [aut, cre],
Garvesh Raskutti [aut]

Maintainer Hyebin Song <`hps5320@psu.edu`>

Repository CRAN

Date/Publication 2026-02-13 09:40:19 UTC

Contents

`PUlasso-package` *PUlasso : An efficient algorithm to solve Positive and Unlabeled(PU) problem with lasso or group lasso penalty*

Description

The package efficiently solves PU problem in low or high dimensional setting using Maximization-Minorization and (block) coordinate descent. It allows simultaneous feature selection and parameter estimation for classification. Sparse calculation and parallel computing are supported for the further computational speed-up. See Hyebin Song, Garvesh Raskutti (2018) <`https://arxiv.org/abs/1711.08129`>.

Details

Main functions: grpPUlasso, cv.grpPUlasso, coef, predict

Author(s)

Hyebin Song, <`hps5320@psu.edu`>, Garvesh Raskutti, <`raskutti@stat.wisc.edu`>.

See Also

Useful links:

- `https://arxiv.org/abs/1711.08129`
- Report bugs at `https://github.com/hsong1/PUlasso/issues`

Examples

```
data("simulPU")
fit<-grpPUlasso(X=simulPU$X,z=simulPU$z,py1=simulPU$truePY1)
## Not run:
cvfit<-cv.grpPUlasso(X=simulPU$X,z=simulPU$z,py1=simulPU$truePY1)

## End(Not run)
coef(fit,lambda=fit$lambda[10])
predict(fit,newdata = head(simulPU$X), lambda= fit$lambda[10],type = "response")
```

cv.grpPUlasso *Cross-validation for PUlasso*

Description

Do a n-fold cross-validation for PUlasso.

Usage

```
cv.grpPUlasso(
  X,
  z,
  py1,
  initial_coef = NULL,
  group = 1:p,
  penalty = NULL,
  lambda = NULL,
  nlambda = 100,
  lambdaMinRatio = ifelse(N < p, 0.05, 0.005),
  maxit = ifelse(method == "CD", 1000, N * 10),
  weights = NULL,
  eps = 1e-04,
  inner_eps = 0.01,
  verbose = FALSE,
  stepSize = NULL,
  stepSizeAdjustment = NULL,
  batchSize = 1,
  updateFrequency = N,
  samplingProbabilities = NULL,
  method = c("CD", "GD", "SGD", "SVRG", "SAG"),
  nfolds = 10,
  fitInd = 1:nfolds,
  nCores = 1,
  trace = c("none", "param", "fVal", "all")
)
```

Arguments

`X`	Input matrix; each row is an observation. Can be a matrix or a sparse matrix.
`z`	Response vector representing whether an observation is labeled or unlabeled.
`py1`	True prevalence Pr(Y=1)
`initial_coef`	A vector representing an initial point where we start PUlasso algorithm from.
`group`	A vector representing grouping of the coefficients. For the least ambiguity, it is recommended if group is provided in the form of vector of consecutive ascending integers.

`penalty`	penalty to be applied to the model. Default is sqrt(group size) for each of the group.
`lambda`	A user supplied sequence of lambda values. If unspecified, the function automatically generates its own lambda sequence based on nlambda and lambdaMinRatio.
`nlambda`	The number of lambda values.
`lambdaMinRatio`	Smallest value for lambda, as a fraction of lambda.max which leads to the intercept only model.
`maxit`	Maximum number of iterations.
`weights`	observation weights. Default is 1 for each observation.
`eps`	Convergence threshold for the outer loop. The algorithm iterates until the maximum change in coefficients is less than eps in the outer loop.
`inner_eps`	Convergence threshold for the inner loop. The algorithm iterates until the maximum change in coefficients is less than eps in the inner loop.
`verbose`	A logical value. if TRUE, the function prints out the fitting process.
`stepSize`	A step size for gradient-based optimization. if NULL, a step size is taken to be stepSizeAdj/mean(Li) where Li is a Lipschitz constant for ith sample
`stepSizeAdjustment`	A step size adjustment. By default, adjustment is 1 for GD and SGD, 1/8 for SVRG and 1/16 for SAG.
`batchSize`	A batch size. Default is 1.
`updateFrequency`	An update frequency of full gradient for method =="SVRG"
`samplingProbabilities`	sampling probabilities for each of samples for stochastic gradient-based optimization. if NULL, each sample is chosen proportionally to Li.
`method`	Optimization method. Default is Coordinate Descent. CD for Coordinate Descent, GD for Gradient Descent, SGD for Stochastic Gradient Descent, SVRG for Stochastic Variance Reduction Gradient, SAG for Stochastic Averaging Gradient.
`nfolds`	Number of cross-validation folds to be created.
`fitInd`	A vector of indices of cross-validation models which will be fitted. Default is to fit the model for each of the cross-validation fold.
`nCores`	Number of threads to be used for parallel computing. If nCores=0, it is set to be (the number of processors available-1) . Default value is 1.
`trace`	An option for saving intermediate quantities when fitting a full dataset.

Value

cvm Mean cross-validation error

cvsd Estimate of standard error of cvm

cvcoef Coefficients for each of the fitted CV models

cvstdcoef Coefficients in a standardized scale for each of the fitted CV models lambda The actual sequence of lambda values used.

lambda.min Value of lambda that gives minimum cvm.

lambda.1se The largest value of lambda such that the error is within 1 standard error of the minimum cvm.

PUfit A fitted PUfit object for the full data

Examples

```
data("simulPU")
fit<-cv.grpPUlasso(X=simulPU$X,z=simulPU$z,py1=simulPU$truePY1)
```

---

deviances *Deviance*

Description

Calculate deviances at provided coefficients

Usage

```
deviances(X, z, py1, coefMat, weights = NULL)
```

Arguments

`X`	Input matrix
`z`	Response vector
`py1`	True prevalence Pr(Y=1)
`coefMat`	A coefficient matrix whose column corresponds to a set of coefficients
`weights`	observation weights. Default is 1 for each observation.

Value

deviances

Examples

```
data("simulPU")
coef0<-replicate(2,runif(ncol(simulPU$X)+1))
deviances(simulPU$X,simulPU$z,py1=simulPU$truePY1,coefMat = coef0)
```

grpPUlasso *Solve PU problem with lasso or group lasso penalty.*

Description

Fit a model using PUlasso algorithm over a regularization path. The regularization path is computed at a grid of values for the regularization parameter lambda.

Usage

```
grpPUlasso(
  X,
  z,
  py1,
  initial_coef = NULL,
  group = 1:ncol(X),
  penalty = NULL,
  lambda = NULL,
  nlambda = 100,
  lambdaMinRatio = ifelse(N < p, 0.05, 0.005),
  maxit = ifelse(method == "CD", 1000, N * 10),
  maxit_inner = 1e+05,
  weights = NULL,
  eps = 1e-04,
  inner_eps = 0.01,
  verbose = FALSE,
  stepSize = NULL,
  stepSizeAdjustment = NULL,
  batchSize = 1,
  updateFrequency = N,
  samplingProbabilities = NULL,
  method = c("CD", "GD", "SGD", "SVRG", "SAG"),
  trace = c("none", "param", "fVal", "all")
)
```

Arguments

`X`	Input matrix; each row is an observation. Can be a matrix or a sparse matrix.
`z`	Response vector representing whether an observation is labeled or unlabeled.
`py1`	True prevalence Pr(Y=1)
`initial_coef`	A vector representing an initial point where we start PUlasso algorithm from.
`group`	A vector representing grouping of the coefficients. For the least ambiguity, it is recommended if group is provided in the form of vector of consecutive ascending integers.
`penalty`	penalty to be applied to the model. Default is sqrt(group size) for each of the group.

`lambda`	A user supplied sequence of lambda values. If unspecified, the function automatically generates its own lambda sequence based on nlambda and lambdaMinRatio.
`nlambda`	The number of lambda values.
`lambdaMinRatio`	Smallest value for lambda, as a fraction of lambda.max which leads to the intercept only model.
`maxit`	Maximum number of iterations.
`maxit_inner`	Maximum number of iterations for a quadratic sub-problem for CD.
`weights`	observation weights. Default is 1 for each observation.
`eps`	Convergence threshold for the outer loop. The algorithm iterates until the maximum change in coefficients is less than eps in the outer loop.
`inner_eps`	Convergence threshold for the inner loop. The algorithm iterates until the maximum change in coefficients is less than eps in the inner loop.
`verbose`	A logical value. if TRUE, the function prints out the fitting process.
`stepSize`	A step size for gradient-based optimization. if NULL, a step size is taken to be stepSizeAdj/mean(Li) where Li is a Lipschitz constant for ith sample
`stepSizeAdjustment`	A step size adjustment. By default, adjustment is 1 for GD and SGD, 1/8 for SVRG and 1/16 for SAG.
`batchSize`	A batch size. Default is 1.
`updateFrequency`	An update frequency of full gradient for method =="SVRG"
`samplingProbabilities`	sampling probabilities for each of samples for stochastic gradient-based optimization. if NULL, each sample is chosen proportionally to Li.
`method`	Optimization method. Default is Coordinate Descent. CD for Coordinate Descent, GD for Gradient Descent, SGD for Stochastic Gradient Descent, SVRG for Stochastic Variance Reduction Gradient, SAG for Stochastic Averaging Gradient.
`trace`	An option for saving intermediate quantities. All intermediate standardized-scale parameter estimates(trace=="param"), objective function values at each iteration(trace=="fVal"), or both(trace=="all") are saved in optResult. Since this is computationally very heavy, it should be only used for decently small-sized dataset and small maxit. A default is "none".

Value

coef A p by length(lambda) matrix of coefficients

std_coef A p by length(lambda) matrix of coefficients in a standardized scale

lambda The actual sequence of lambda values used.

nullDev Null deviance defined to be 2*(logLik_sat -logLik_null)

deviance Deviance defined to be 2*(logLik_sat -logLik(model)) optResult A list containing the result of the optimization. fValues, subGradients contain objective function values and subgradient vectors at each lambda value. If trace = TRUE, corresponding intermediate quantities are saved as well.

iters Number of iterations(EM updates) if method = "CD". Number of steps taken otherwise.

Examples

```
data("simulPU")
fit<-grpPUlasso(X=simulPU$X,z=simulPU$z,py1=simulPU$truePY1)
```

`simulPU` *simulated PU data*

Description

A simulated data for the illustration. Covariates x_i are drawn from $N(\mu, I_{5\times5})$ or $N(-\mu, I_{5\times5})$ with probability 0.5. To make the first two variables active,$\mu = [\mu_1, \ldots, \mu_2, 0, 0, 0]^T, \theta = [\theta_0, \ldots, \theta_2, 0, 0, 0]^T$ and we set $\mu_i = 1.5, \theta_i \sim Unif[0.5, 1]$ Responses y_i is simulated via $P_\theta(y = 1|x) = 1/exp(-\theta^T x)$. 1000 observations are sampled from the sub-population of positives(y=1) and labeled, and another 1000 observations are sampled from the original population and unlabeled.

Usage

```
data('simulPU')
```

Format

A list containing model matrix X, true response y, labeled/unlabeled response vector z, and a true positive probability truePY1.

Index