Package ‘milr’

October 13, 2022

Type Package

Title Multiple-Instance Logistic Regression with LASSO Penalty

Version 0.3.1

Date 2020-10-28

Description The multiple instance data set consists of many independent subjects (called bags) and each subject is composed of several components (called instances). The outcomes of such data set are binary or categorical responses, and, we can only observe the subject-level outcomes. For example, in manufacturing processes, a subject is labeled as ``defective'' if at least one of its own components is defective, and otherwise, is labeled as ``non-defective''. The 'milr' package focuses on the predictive model for the multiple instance data set with binary outcomes and performs the maximum likelihood estimation with the Expectation-Maximization algorithm under the framework of logistic regression. Moreover, the LASSO penalty is attached to the likelihood function for simultaneous parameter estimation and variable selection.

Author Ping-Yang Chen [aut, cre],
ChingChuan Chen [aut],
Chun-Hao Yang [aut],
Sheng-Mao Chang [aut]

URL https://github.com/PingYangChen/milr

BugReports https://github.com/PingYangChen/milr/issues

Maintainer Ping-Yang Chen <pychen.ping@gmail.com>

Depends R (>= 3.2.3)

Imports utils, pipeR (>= 0.5), numDeriv, glmnet, Rcpp (>= 0.12.0), RcppParallel

LinkingTo Rcpp, RcppArmadillo, RcppParallel

Suggests testthat, knitr, Hmisc, rmarkdown, data.table, ggplot2, plyr

LazyData yes

SystemRequirements GNU make

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-10-31 07:30:02 UTC

R topics documented:

`milr-package` *The milr package: multiple-instance logistic regression with lasso penalty*

Description

The multiple instance data set consists of many independent subjects (called bags) and each subject is composed of several components (called instances). The outcomes of such data set are binary or multinomial, and, we can only observe the subject-level outcomes. For example, in manufactory processes, a subject is labeled as "defective" if at least one of its own components is defective, and otherwise, is labeled as "non-defective". The milr package focuses on the predictive model for the multiple instance data set with binary outcomes and performs the maximum likelihood estimation with the Expectation-Maximization algorithm under the framework of logistic regression. Moreover, the LASSO penalty is attached to the likelihood function for simultaneous parameter estimation and variable selection.

References

1. Chen, R.-B., Cheng, K.-H., Chang, S.-M., Jeng, S.-L., Chen, P.-Y., Yang, C.-H., and Hsia, C.-C. (2016). Multiple-Instance Logistic Regression with LASSO Penalty. arXiv:1607.03615 [stat.ML].

DGP *DGP: data generation*

Description

Generating the multiple-instance data set.

Usage

```
DGP(n, m, beta)
```

Arguments

`n`	an integer. The number of bags.
`m`	an integer or vector of length `n`. If `m` is an integer, each bag has the identical number of instances, `m`. If `m` is a vector, the `i`th bag has `m[i]` instances.
`beta`	a vector. The true regression coefficients.

Value

a list including (1) bag-level labels, `Z`, (2) the design matrix, `X`, and (3) bag ID of each instance, `ID`.

Examples

```
data1 <- DGP(50, 3, runif(10, -5, 5))
data2 <- DGP(50, sample(3:5, 50, TRUE), runif(10, -5, 5))
```

fitted.milr *Fitted Response of milr Fits*

Description

Fitted Response of milr Fits

Usage

```
## S3 method for class 'milr'
fitted(object, type = "bag", ...)
```

Arguments

`object`	A fitted obejct of class inheriting from `"milr"`.
`type`	The type of fitted response required. Default is `"bag"`, the fitted labels of bags. The `"instance"` option returns the fitted labels of instances.
`...`	further arguments passed to or from other methods.

`fitted.softmax` *Fitted Response of softmax Fits*

Description

Fitted Response of softmax Fits

Usage

```
## S3 method for class 'softmax'
fitted(object, type = "bag", ...)
```

Arguments

`object`	A fitted obejct of class inheriting from `"softmax"`.
`type`	The type of fitted response required. Default is `"bag"`, the fitted labels of bags. The `"instance"` option returns the fitted labels of instances.
`...`	further arguments passed to or from other methods.

`logit` *logit link function*

Description

calculate the values of logit link

Usage

```
logit(X, beta)
```

Arguments

`X`	A matrix, the design matrix.
`beta`	A vector, the coefficients.

Value

An vector of the values of logit link.

milr *Maximum likelihood estimation of multiple-instance logistic regression with LASSO penalty*

Description

Please refer to milr-package.

Usage

```
milr(
  y,
  x,
  bag,
  lambda = 0,
  numLambda = 20L,
  lambdaCriterion = "BIC",
  nfold = 10L,
  maxit = 500L
)
```

Arguments

`y`	a vector. Bag-level binary labels.
`x`	the design matrix. The number of rows of `x` must be equal to the length of `y`.
`bag`	a vector, bag id.
`lambda`	the tuning parameter for LASSO-penalty. If `lambda` is a real value number, then the `milr` fits the model based on this lambda value. Second, if `lambda` is vector, then the optimal lambda value would be be chosen based on the optimality criterion, `lambdaCriterion`. Finally, if `lambda = -1`, then the optimal lambda value would be chosen automatically. The default is 0.
`numLambda`	An integer, the maximum length of LASSO-penalty. in atuo-tunning mode (`lambda = -1`). The default is 20.
`lambdaCriterion`	a string, the used optimality criterion for tuning the `lambda` value. It can be specified with `lambdaCriterion = "BIC"` or `lambdaCriterion = "deviance"`.
`nfold`	an integer, the number of fold for cross-validation to choose the optimal `lambda` when `lambdaCriterion = "deviance"`.
`maxit`	an integer, the maximum iteration for the EM algorithm. The default is 500.

Value

An object with S3 class "milr".

- lambdaa vector of candidate lambda values.

- cva vector of predictive deviance via `nfold`-fold cross validation when `lambdaCriterion = "deviance"`.
- deviancea vector of deviance of candidate model for each candidate lambda value.
- BICa vector of BIC of candidate model for each candidate lambda value.
- best_indexan integer, indicates the index of the best model among candidate lambda values.
- best_modela list of the information for the best model including deviance (not cv deviance), BIC, chosen lambda, coefficients, fitted values, log-likelihood and variances of coefficients.

Examples

```
set.seed(100)
beta <- runif(5, -5, 5)
trainData <- DGP(40, 3, beta)
testData <- DGP(5, 3, beta)
# default (not use LASSO)
milr_result <- milr(trainData$Z, trainData$X, trainData$ID)
coef(milr_result)      # coefficients
fitted(milr_result)                    # fitted bag labels
fitted(milr_result, type = "instance") # fitted instance labels
summary(milr_result)   # summary milr
predict(milr_result, testData$X, testData$ID)                    # predicted bag labels
predict(milr_result, testData$X, testData$ID, type = "instance") # predicted instance labels

# use BIC to choose penalty (not run)
#milr_result <- milr(trainData$Z, trainData$X, trainData$ID,
#                    exp(seq(log(0.01), log(50), length = 30)))
#coef(milr_result)      # coefficients
#fitted(milr_result)                    # fitted bag labels
#fitted(milr_result, type = "instance") # fitted instance labels
#summary(milr_result)   # summary milr
#predict(milr_result, testData$X, testData$ID)                    # predicted bag labels
#predict(milr_result, testData$X, testData$ID, type = "instance") # predicted instance labels

# use auto-tuning (not run)
#milr_result <- milr(trainData$Z, trainData$X, trainData$ID, lambda = -1, numLambda = 20)
#coef(milr_result)      # coefficients
#fitted(milr_result)                    # fitted bag labels
#fitted(milr_result, type = "instance") # fitted instance labels
#summary(milr_result)   # summary milr
#predict(milr_result, testData$X, testData$ID)                    # predicted bag labels
#predict(milr_result, testData$X, testData$ID, type = "instance") # predicted instance labels

# use cv in auto-tuning (not run)
#milr_result <- milr(trainData$Z, trainData$X, trainData$ID,
#                    lambda = -1, numLambda = 20, lambdaCriterion = "deviance")
#coef(milr_result)      # coefficients
#fitted(milr_result)                    # fitted bag labels
#fitted(milr_result, type = "instance") # fitted instance labels
#summary(milr_result)   # summary milr
#predict(milr_result, testData$X, testData$ID)                    # predicted bag labels
#predict(milr_result, testData$X, testData$ID, type = "instance") # predicted instance labels
```

predict.milr *Predict Method for milr Fits*

Description

Predict Method for milr Fits

Usage

```
## S3 method for class 'milr'
predict(object, newdata = NULL, bag_newdata = NULL, type = "bag", ...)
```

Arguments

`object`	A fitted obejct of class inheriting from `"milr"`.
`newdata`	Default is `NULL`. A matrix with variables to predict.
`bag_newdata`	Default is `NULL`. A vector. The labels of instances to bags. If `newdata` and `bag_newdata` both are `NULL`, return the fitted result.
`type`	The type of prediction required. Default is `"bag"`, the predicted labels of bags. The `"instance"` option returns the predicted labels of instances.
`...`	further arguments passed to or from other methods.

predict.softmax *Predict Method for softmax Fits*

Description

Predict Method for softmax Fits

Usage

```
## S3 method for class 'softmax'
predict(object, newdata = NULL, bag_newdata = NULL, type = "bag", ...)
```

Arguments

`object`	A fitted obejct of class inheriting from `"softmax"`.
`newdata`	Default is `NULL`. A matrix with variables to predict.
`bag_newdata`	Default is `NULL`. A vector. The labels of instances to bags. If `newdata` and `bag_newdata` both are `NULL`, return the fitted result.
`type`	The type of prediction required. Default is `"bag"`, the predicted labels of bags. The `"instance"` option returns the predicted labels of instances.
`...`	further arguments passed to or from other methods.

softmax *Multiple-instance logistic regression via softmax function*

Description

This function calculates the alternative maximum likelihood estimation for multiple-instance logistic regression through a softmax function (Xu and Frank, 2004; Ray and Craven, 2005).

Usage

```
softmax(y, x, bag, alpha = 0, ...)
```

Arguments

`y`	a vector. Bag-level binary labels.
`x`	the design matrix. The number of rows of `x` must be equal to the length of `y`.
`bag`	a vector, bag id.
`alpha`	A non-negative realnumber, the softmax parameter.
`...`	arguments to be passed to the `optim` function.

Value

a list including coefficients and fitted values.

References

1. S. Ray, and M. Craven. (2005) Supervised versus multiple instance learning: An empirical comparsion. in Proceedings of the 22nd International Conference on Machine Learnings, ACM, 697–704.
2. X. Xu, and E. Frank. (2004) Logistic regression and boosting for labeled bags of instances. in Advances in Knowledge Discovery and Data Mining, Springer, 272–281.

Examples

```
set.seed(100)
beta <- runif(10, -5, 5)
trainData <- DGP(40, 3, beta)
testData <- DGP(5, 3, beta)
# Fit softmax-MILR model S(0)
softmax_result <- softmax(trainData$Z, trainData$X, trainData$ID, alpha = 0)
coef(softmax_result)      # coefficients
fitted(softmax_result)                    # fitted bag labels
fitted(softmax_result, type = "instance") # fitted instance labels
predict(softmax_result, testData$X, testData$ID)                  # predicted bag labels
predict(softmax_result, testData$X, testData$ID, type = "instance") # predicted instance labels
# Fit softmax-MILR model S(3) (not run)
# softmax_result <- softmax(trainData$Z, trainData$X, trainData$ID, alpha = 3)
```

Index