Package 'MVB'

February 19, 2015

Type Package

Title Mutivariate Bernoulli log-linear model

Version 1.1

Date 2012-12-29

Author Bin Dai

Maintainer Bin Dai <dai@stat.wisc.edu>

Description Fit log-linear model for multivariate Bernoulli distribution with mixed effect models and LASSO

License GPL (>= 2.0)

Depends Rcpp (>= 0.9.9), RcppArmadillo (>= 0.2.34)

LinkingTo Rcpp, RcppArmadillo

Repository CRAN

Date/Publication 2013-12-15 11:24:08

NeedsCompilation yes

R topics documented:

MVB-package *MVB as Multivariate Bernoulli*

Description

Functionality for multivairate Bernoulli distribution including log-linear models, lasso variable selection and mixed effects models.

Details

Package:	MVB
Type:	Package
Version:	1.0
Date:	2012-03-21
License:	GPL (>=2)

Author(s)

Bin Dai <daibin at stat dot wisc dot edu>

Examples

```
# fit a simple MVB log-linear model
n <- 1000
p <- 5
kk <- 2
tt <- NULL
alter <- 1
for (i in 1:kk) {
  vec <- rep(0, p)
  vec[i] <- alter
  alter <- alter * (-1)
  tt <- cbind(tt, vec)
}
tt <- 1.5 * tt
tt <- cbind(tt, c(rep(0, p - 1), 1))

x <- matrix(rnorm(n * p, 0, 4), n, p)
res <- mvb.simu(tt, x, K = kk, rep(.5, 2))
fitMVB <- mvbfit(x, res$response, output = 1)
```

loglike *negative loglikelihood evaluation*

Description

evaluate negative loglikelihood of the corresponding family of model.

Usage

```
loglike(x, y, input,
        family = c("gaussian", "bernoulli", "mvbernoulli"))
```

Arguments

x	design matrix.
y	output binary matrix with number of columns equal to the number of outcomes per observation.
input	vector of the fitted coefficients for the distribution family.
family	a GLM family, currently support gaussian, binomial and mvbernoulli (multivariate Bernoulli).

Details

evaluate the negative log-likelihood to examine the performance of the model.

Value

a double value returned as the negative log-likelihood

See Also

unifit, mvbfit

Examples

```
# fit a simple MVB log-linear model
n <- 1000
p <- 5
kk <- 2
tt <- NULL
alter <- 1
for (i in 1:kk) {
  vec <- rep(0, p)
  vec[i] <- alter
  alter <- alter * (-1)
  tt <- cbind(tt, vec)
}
tt <- 1.5 * tt
```

```
tt <- cbind(tt, c(rep(0, p - 1), 1))

x <- matrix(rnorm(n * p, 0, 4), n, p)
res <- mvb.simu(tt, x, K = kk, rep(.5, 2))
fitMVB <- mvbfit(x, res$response, output = 1)
loglike(x, res$response, fitMVB$beta, "mvbernoulli")
```

`mvb.simu` *generate multivariate Bernoulli simulated data*

Description

for given coefficients and design matrix, generate the corresponding responses according multivariate Bernoulli model

Usage

```
mvb.simu(coefficients, x, K = 2, offset = as.double(0))
```

Arguments

`coefficients`	coefficients matrix, number of columns should be less than `2^K`.
`x`	design matrix.
`K`	number of outcomes for the model.
`offset`	non-penalized terms in coefficients, corresponding to a unit column in design matrix, which is generated automaticly.

Details

The response variables are simulated according to cononical link function of multivariate Bernoulli model with coefficients speicified.

Value

`response`	matrix for outcomes, with dimension `nobs` times `K`.
`beta`	expanded coefficients from input argument `coefficients` and `offset`.

See Also

`mvbfit`, `mvblps`

Examples

```
# fit a simple MVB log-linear model
n <- 1000
p <- 5
kk <- 2
tt <- NULL
alter <- 1
for (i in 1:kk) {
  vec <- rep(0, p)
  vec[i] <- alter
  alter <- alter * (-1)
  tt <- cbind(tt, vec)
}
tt <- 1.5 * tt
tt <- cbind(tt, c(rep(0, p - 1), 1))

x <- matrix(rnorm(n * p, 0, 4), n, p)
res <- mvb.simu(tt, x, K = kk, rep(.5, 2))
fitMVB <- mvbfit(x, res$response, output = 1)
```

`mvbfit` *multivariate Bernoulli logistic model fitting*

Description

fit multivariate Bernoulli logistic model using Newton-Raphson algorithm.

Usage

```
mvbfit(x, y, maxOrder = 2,
       output = 0, printIter = 100)
```

Arguments

`x`	input design matrix.
`y`	output binary matrix with number of columns equal to the number of outcomes per observation.
`maxOrder`	maximum order of interactions to be considered in outcomes.
`output`	with values 0 or 1, indicating whether the fitting process is muted or not.
`printIter`	Number of iterations to be printed if output is true.

Details

The `mvbfit` utilize the class structure of the underlying C++ code and fitted the model with Newton-Raphson algorithm.

Value

An object of class `mvbfit`, for which some methods are available.

See Also

`mvblps`, `unifit`, `stepfit`, `mvb.simu`

Examples

```
# fit a simple MVB log-linear model
n <- 1000
p <- 5
kk <- 2
tt <- NULL
alter <- 1
for (i in 1:kk) {
  vec <- rep(0, p)
  vec[i] <- alter
  alter <- alter * (-1)
  tt <- cbind(tt, vec)
}
tt <- 1.5 * tt
tt <- cbind(tt, c(rep(0, p - 1), 1))

x <- matrix(rnorm(n * p, 0, 4), n, p)
res <- mvb.simu(tt, x, K = kk, rep(.5, 2))
fitMVB <- mvbfit(x, res$response, output = 1)
```

---

mvblps *multivariate Bernoulli LASSO model fitting*

Description

fit multivariate Bernoulli LASSO model accelerated block-coordinate relaxation algorithm.

Usage

```
mvblps(x, y, maxOrder = 2, lambda = NULL, nlambda = 100,
       lambda.min.ratio = ifelse(nobs<nvars, .01, .0001),
       output = 0, printIter = 100, search = c('nm', 'grid'),
       tune = c("AIC", "BIC", "GACV", "BGACV"))
```

Arguments

x	input design matrix.
y	output binary matrix with number of columns equal to the number of outcomes per observation.

maxOrder	maximum order of interactions to be considered in outcomes.
lambda	a user specified tuning sequece. Typical usage is to have the program compute its own lambda.
nlambda	the number of lambda values, default is 100.
lambda.min.ratio	Smallest value for lambda, as a fraction of lambda.max. The default depends on the sample size nobs relative to the number of variables.
output	with values 0 or 1, indicating whether the fitting process is muted or not.
printIter	Number of iterations to be printed if output is true.
search	Tuning search approach, nm for Nelder Mead and grid for grid search.
tune	tuning approach, available methods including AIC, BIC, GACV, BGACV.

Details

The mvblps utilize the class structure of the underlying C++ code and fitted the model with accelerated block-coordinate relaxation algorithm.

Value

An object of classes mvbfit and lps, for which some methods are available.

See Also

mvbfit, unifit, stepfit, mvb.simu

Examples

```
# fit a simple MVB log-linear model
n <- 1000
p <- 5
kk <- 2
tt <- NULL
alter <- 1
for (i in 1:kk) {
  vec <- rep(0, p)
  vec[i] <- alter
  alter <- alter * (-1)
  tt <- cbind(tt, vec)
}
tt <- 1.5 * tt
tt <- cbind(tt, c(rep(0, p - 1), 1))

x <- matrix(rnorm(n * p, 0, 4), n, p)
res <- mvb.simu(tt, x, K = kk, rep(.5, 2))
fitMVB <- mvblps(x, res$response, output = 1)
```

mvbme *multivariate Bernoulli mixed-effects model fitting*

Description

fit multivariate Bernoulli mixed-effects model using Laplacian approximation.

Usage

```
mvbme(x, y, z, maxOrder = 2,
       output = 0, printIter = 100)
```

Arguments

x	input design matrix.
y	output binary matrix with number of columns equal to the number of outcomes per observation.
z	random effect design matrix.
maxOrder	maximum order of interactions to be considered in outcomes.
output	with values 0 or 1, indicating whether the fitting process is muted or not.
printIter	Number of iterations to be printed if output is true.

Details

The mvbme utilize the class structure of the underlying C++ code and fitted the model with Laplacian approximation.

Value

An object of class mvbfit, for which some methods are available.

See Also

mvblps, unifit, stepfit, mvb.simu

Examples

```
# fit a simple MVB log-linear model
n <- 1000
p <- 5
kk <- 2
tt <- NULL
alter <- 1
for (i in 1:kk) {
  vec <- rep(0, p)
  vec[i] <- alter
  alter <- alter * (-1)
```

```
  tt <- cbind(tt, vec)
}
tt <- 1.5 * tt
tt <- cbind(tt, c(rep(0, p - 1), 1))

x <- matrix(rnorm(n * p, 0, 4), n, p)
res <- mvb.simu(tt, x, K = kk, rep(.5, 2))
fitMVB <- mvbfit(x, res$response, output = 1)
```

`stepfit` *step-wisd multivariate model fitting*

Description

stepwise fit multivariate log-linear Bernoulli model using Newton-Raphson algorithm.

Usage

```
stepfit(x, y, maxOrder = 2,
        output = 0,
        direction = c("backward", "forward"),
        tune = c("AIC", "BIC", "GACV", "BGACV"),
        start = NULL)
```

Arguments

`x`	input design matrix.
`y`	output binary matrix with number of columns equal to the number of outcomes per observation.
`maxOrder`	maximum order of interactions to be considered in outcomes.
`output`	with values 0 or 1, indicating whether the fitting process is muted or not.
`direction`	the mode of stepwise search and default is backward.
`tune`	tuning approach, available methods including AIC, BIC, GACV, BGACV.
`start`	starting object of type mvbfit.

Details

The `stepfit` utilize the class structure of the underlying C++ code and stepwisd fitted the model with Newton-Raphson algorithm.

Value

An object of class `mvbfit`, for which some methods are available.

See Also

mvblps, unifit, stepfit, mvb.simu

Examples

```
# fit a simple MVB log-linear model
n <- 1000
p <- 5
kk <- 2
tt <- NULL
alter <- 1
for (i in 1:kk) {
  vec <- rep(0, p)
  vec[i] <- alter
  alter <- alter * (-1)
  tt <- cbind(tt, vec)
}
tt <- 1.5 * tt
tt <- cbind(tt, c(rep(0, p - 1), 1))

x <- matrix(rnorm(n * p, 0, 4), n, p)
res <- mvb.simu(tt, x, K = kk, rep(.5, 2))
fitMVB <- mvbfit(x, res$response, output = 1)
```

---

unifit *univariate model fitting*

Description

fit univariate log-linear model using Newton-Raphson algorithm.

Usage

```
unifit(formula, data = list(),
       family = c("gaussian", "binomial"),
       output = 0)
```

Arguments

formula	a symbolic description of the model to be fit.
data	an optional data frame containing the variables in the model. By default the variables are taken from the environment from which unifit is called.
family	a GLM family, currently support gaussian and binomial.
output	with values 0 or 1, indicating whether the fitting process is muted or not.

Details

The `unifit` utilize the class structure of the underlying C++ code and fitted the model with Newton-Raphson algorithm.

Value

An object of class `mvbfit`, for which some methods are available.

See Also

`unilps`, `mvbfit`

Examples

```
n <- 100
p <- 4
x <- matrix(rnorm(n * p, 0, 4), n, p)
eta <- x
pr <- exp(eta) / (1+ exp(eta))
res <- rbinom(n, 1, pr)
fit <- unifit(res ~ x - 1, family = 'binomial')
```

---

unilps *univariate model fitting with lasso penalty*

Description

fit univariate log-linear model using accelerated block-coordinate relaxation algorithm.

Usage

```
unilps(formula, data = list(),
       family = c("gaussian", "binomial"),
       lambda = NULL, nlambda = 100,
       lambda.min.ratio = ifelse(nobs<nvars, .01, .0001),
       output = 0, tune = c("AIC", "BIC", "GACV", "BGACV"))
```

Arguments

`formula`	a symbolic description of the model to be fit.
`data`	an optional data frame containing the variables in the model. By default the variables are taken from the environment from which `unifit` is called.
`family`	a GLM family, currently suport gaussian and binomial.
`lambda`	a user specified tuning sequece. Typical usage is to have the program compute its own `lambda`.
`nlambda`	the number of `lambda` values, default is 100.

`lambda.min.ratio`
: Smallest value for `lambda`, as a fraction of `lambda.max`. The default depends on the sample size `nobs` relative to the number of variables.

`output`
: with values 0 or 1, indicating whether the fitting process is muted or not.

`tune`
: tuning approach, available methods including AIC, BIC, GACV, BGACV.

Details

The `unilps` utilize the class structure of the underlying C++ code and fitted the model with accelerated block-coordinate relaxation algorithm.

Value

An object of classes `mvbfit` and `lps`, for which some methods are available.

See Also

`unilps`, `mvblps`

Examples

```
n <- 100
p <- 4
x <- matrix(rnorm(n * p, 0, 4), n, p)
eta <- x
pr <- exp(eta) / (1+ exp(eta))
res <- rbinom(n, 1, pr)
fit <- unilps(res ~ x - 1, family = 'binomial')
```

Index