Package 'npmr'

July 22, 2025

Type Package

Title Nuclear Penalized Multinomial Regression

Version 1.3.1

Date 2023-11-11

Author Scott Powers, Trevor Hastie, Robert Tibshirani

Maintainer Scott Powers <saberpowers@gmail.com>

Description Fit multinomial logistic regression with a penalty on the nuclear norm of the estimated regression coefficient matrix, using proximal gradient descent.

Imports Matrix

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2023-11-11 18:13:18 UTC

Contents

---

npmr-package *Nuclear penalized multinomial regression*

Description

As an alternative to an l1- or l2-penalty on multinomial logistic regression, this package fits multinomial regression with a penalty on the nuclear norm of the fitted regression coefficient matrix. The result is often a matrix of reduced rank, leveraging structure among the response classes so that the likelihood of one class informs the likelihood of other classes. Proximal gradient descent is used to solve the NPMR optimization problem.

Details

The primary functions in the package are `npmr`, which solves nuclear penalized multinomial regression for a sequence of input values for the regularization parameter lambda, and `cv.npmr`, which chooses the optimal value of the regularization parameter lambda via cross validation. Both `npmr` and `cv.npmr` have predict and plot methods.

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

Maintainer: Scott Powers <sspowers@stanford.edu>

References

Scott Powers, Trevor Hastie and Rob Tibshirani (2016). "Nuclear penalized multinomial regression with an application to predicting at bat outcomes in baseball." In prep.

Examples

```
#   Fit NPMR to simulated data

K = 5
n = 1000
m = 10000
p = 10
r = 2

# Simulated training data
set.seed(8369)
A = matrix(rnorm(p*r), p, r)
C = matrix(rnorm(K*r), K, r)
B = tcrossprod(A, C)            # low-rank coefficient matrix
X = matrix(rnorm(n*p), n, p)    # covariate matrix with iid Gaussian entries
eta = X
P = exp(eta)/rowSums(exp(eta))
Y = t(apply(P, 1, rmultinom, n = 1, size = 1))
```

```
# Simulate test data
Xtest = matrix(rnorm(m*p), m, p)
etatest = Xtest
Ptest = exp(etatest)/rowSums(exp(etatest))
Ytest = t(apply(Ptest, 1, rmultinom, n = 1, size = 1))

# Fit NPMR for a sequence of lambda values without CV:
fit2 = npmr(X, Y, lambda = exp(seq(7, -2)))

# Print the NPMR fit:
fit2

# Produce a biplot:
plot(fit2, lambda = 20)

# Compute mean test error using the predict function (for each value of lambda):
getloss = function(pred, Y) {
    -mean(log(rowSums(Y*pred)))
}
apply(predict(fit2, Xtest), 3, getloss, Ytest)
```

`cv.npmr` *Cross-validated nuclear penalized multinomial regression*

Description

Divide the training data into folds. Hold out each fold and fit NPMR for a range of regularization values on the remaining data, testing the result on the held-out fold. After the optimal value of the regularization parameter is determined, fit NPMR with this tuning parameter to the whole training set.

Usage

```
cv.npmr(X, Y, lambda = exp(seq(7, -2)), s = 0.1/max(X), eps = 1e-06,
    group = NULL, accelerated = TRUE, B.init = NULL, b.init = NULL,
    foldid = NULL, nfolds = 10)
```

Arguments

`X`	Covariate matrix. May be in sparse form from `Matrix` package
`Y`	Multinomial reponse. May be (1) a vector of length equal to nrow(X), which will be interpreted as a factor, with levels representing response classes; or (2) a matrix with nrow(Y) = nrow(X), and each row has exactly one 1 representing the response class for that observation, with the remaining entries of the row being zero.

lambda	Vector of regularization parameter values for penalizing nuclear norm. Default is a wide range of values. We suggest that the user choose this sequence via trial and error. If the model takes too long to fit, try larger values of lambda. If cross validation error is strictly increasing or strictly decreasing over the range of lambda specified, try extending the range in the direction of the smallest cross validation error.
s	Step size for proximal gradient descent
eps	Convergence threshold. When relative change in the objective function after an interation drops below this threshold, algorithm halts.
group	Vector of length equal to number of variables (ncol(X) and nrow(B)). Variables in the same group indexed by a POSITIVE integer will be penalized together (the nuclear norm of the sub-matrix of the regression coefficients will be penalized). Variables without positive integers will NOT be penalized. Default is NULL, which means there are no sub-groups; nuclear norm of entire coefficient matrix is penalized.
accelerated	Logical. Should accelerated proximal gradient descent be used? Default is TRUE.
B.init	Initial value of the regression coefficient matrix for proximal gradient descent
b.init	Initial value of the regression intercept vector for proximal gradient descent
foldid	Vector of length equal to nrow(X). Specifies folds for cross validation.
nfolds	Number of folds for cross validation. Ignored if `foldid` is specified. Default is 10.

Value

An object of class "cv.npmr" with values:

call	the call that produced this object
error	A vector of total cross validation error for each value of `lambda`
fit	An object of class `npmr` fitted to the entire training data using `lambda.min`
lambda.min	The value of lambda with minimum cross validation error
lambda	The input sequence of regularization parameter values for which cross validation error was calculated
n	number of rows in the input covariate matrix `X`

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

References

Scott Powers, Trevor Hastie and Rob Tibshirani (2016). "Nuclear penalized multinomial regression with an application to predicting at bat outcomes in baseball." In prep.

See Also

`npmr`, `predict.cv.npmr`, `print.cv.npmr`, `plot.cv.npmr`

Examples

```
#   Fit NPMR to simulated data

K = 5
n = 1000
m = 10000
p = 10
r = 2

# Simulated training data
set.seed(8369)
A = matrix(rnorm(p*r), p, r)
C = matrix(rnorm(K*r), K, r)
B = tcrossprod(A, C)            # low-rank coefficient matrix
X = matrix(rnorm(n*p), n, p)    # covariate matrix with iid Gaussian entries
eta = X
P = exp(eta)/rowSums(exp(eta))
Y = t(apply(P, 1, rmultinom, n = 1, size = 1))
fold = sample(rep(1:10, length = nrow(X)))

# Simulate test data
Xtest = matrix(rnorm(m*p), m, p)
etatest = Xtest
Ptest = exp(etatest)/rowSums(exp(etatest))
Ytest = t(apply(Ptest, 1, rmultinom, n = 1, size = 1))

# Fit NPMR for a sequence of lambda values without CV:
fit2 = cv.npmr(X, Y, lambda = exp(seq(7, -2)), foldid = fold)

# Print the NPMR fit:
fit2

# Produce a biplot:
plot(fit2)

# Compute mean test error using the predict function:
-mean(log(rowSums(Ytest*predict(fit2, Xtest))))
```

---

logL *Log-likelihood for multinomial regression model*

Description

Computes the log-likelihood of the fitted regression parameters given the data observed. Intended for internal use only.

Usage

```
logL(B, b, X, Y)
```

Arguments

B	Regression coefficient matrix
b	Regression intercept vector
X	Covariate matrix
Y	Multinomial response matrix

Value

The log-likelihood of B and b given X and Y

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

See Also

nuclear, objective

---

npmr *Nuclear penalized multinomial regression*

Description

Fit a multinomial logistic regression model for a sequence of regularization parameters penalizing the nuclear norm of the regression coefficient matrix.

Usage

```
npmr(X, Y, lambda = exp(seq(7, -2)), s = 0.1/max(X), eps = 1e-06, group = NULL,
    accelerated = TRUE, B.init = NULL, b.init = NULL, quiet = TRUE)
```

Arguments

X	Covariate matrix. May be in sparse form from Matrix package
Y	Multinomial reponse. May be (1) a vector of length equal to nrow(X), which will be interpreted as a factor, with levels representing response classes; or (2) a matrix with nrow(Y) = nrow(X), and each row has exactly one 1 representing the response class for that observation, with the remaining entries of the row being zero.
lambda	Vector of regularization parameter values for penalizing nuclear norm. Default is a wide range of values. We suggest that the user choose this sequence via trial and error. If the model takes too long to fit, try larger values of lambda.
s	Step size for proximal gradient descent
eps	Convergence threshold. When relative change in the objective function after an interation drops below this threshold, algorithm halts.

group	Vector of length equal to number of variables (ncol(X) and nrow(B)). Variables in the same group indexed by a POSITIVE integer will be penalized together (the nuclear norm of the sub-matrix of the regression coefficients will be penalized). Variables without positive integers will NOT be penalized. Default is NULL, which means there are no sub-groups; nuclear norm of entire coefficient matrix is penalized.
accelerated	Logical. Should accelerated proximal gradient descent be used? Default is TRUE.
B.init	Initial value of the regression coefficient matrix for proximal gradient descent
b.init	Initial value of the regression intercept vector for proximal gradient descent
quiet	Logical. Should output be silenced? If not, print the value of the objective function after each step of proximal gradient descent. Perhaps useful for debugging. Default is TRUE.

Details

In multinomial regression (in contrast with Gaussian regression or logistic regression) there is a matrix of regression coefficients, not just a vector. NPMR fits a logistic multinomial regression with a penalty on the nuclear norm of this regression coefficient matrix B. Specifically, the objective is

-loglik(B,b|X,Y) + lambda*||B||_*

where ||.||_* denotes the nuclear norm. This implementation solves the problem using proximal gradient descent, which iteratively steps in the direction of the negative gradient of the loss function and soft-thresholds the singular values of the result.

This function makes available the option, through the groups argument, of dividing the regression coefficient matrix into sub-matrices (by row) and penalizing the sum of the nuclear norms of these submatrices. Rows (correspond to variables) can be given no penalty in this way.

Value

An object of class "npmr" with values:

call	the call that produced this object
B	A 3-dimensional array, with dimensions (ncol(X), ncol(Y), length(lambda)). For each lambda, this array stores the regression coefficient matrix which solves the NPMR optimization problem for that value of lambda.
b	A matrix with ncol(Y) rows and length(lambda) columns. Each column stores the regression intercept vector which solves the NPMR optimization problem for that value of lambda.
objective	A vector of length equal to the length of lambda, giving the value value of the objective for the solution corresponding to each value of lambda.
lambda	The input sequence of values for the regularization parameter, for each of which NPMR has been solved.

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

References

Scott Powers, Trevor Hastie and Rob Tibshirani (2016). "Nuclear penalized multinomial regression with an application to predicting at bat outcomes in baseball." In prep.

See Also

cv.npmr, predict.npmr, print.npmr, plot.npmr

Examples

```
#   Fit NPMR to simulated data

K = 5
n = 1000
m = 10000
p = 10
r = 2

# Simulated training data
set.seed(8369)
A = matrix(rnorm(p*r), p, r)
C = matrix(rnorm(K*r), K, r)
B = tcrossprod(A, C)            # low-rank coefficient matrix
X = matrix(rnorm(n*p), n, p)    # covariate matrix with iid Gaussian entries
eta = X
P = exp(eta)/rowSums(exp(eta))
Y = t(apply(P, 1, rmultinom, n = 1, size = 1))

# Simulate test data
Xtest = matrix(rnorm(m*p), m, p)
etatest = Xtest
Ptest = exp(etatest)/rowSums(exp(etatest))
Ytest = t(apply(Ptest, 1, rmultinom, n = 1, size = 1))

# Fit NPMR for a sequence of lambda values without CV:
fit2 = npmr(X, Y, lambda = exp(seq(7, -2)))

# Print the NPMR fit:
fit2

# Produce a biplot:
plot(fit2, lambda = 20)

# Compute mean test error using the predict function (for each value of lambda):
getloss = function(pred, Y) {
    -mean(log(rowSums(Y*pred)))
}
apply(predict(fit2, Xtest), 3, getloss, Ytest)
```

nuclear *Nuclear norm of a matrix*

Description

Returns the nuclear norm of a matrix, which is the sum of its singular values, obtained through a singular value decomposition. Intended for internal use only.

Usage

```
nuclear(B)
```

Arguments

B	a matrix

Value

the nuclear norm of the matrix

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

See Also

logL, objective, objectiveFast

objective *NPMR objective function*

Description

Return the objective function of the data and the fitted parameters for nuclear penalized multinomial regression. The objective is the sum of the negative log-likelihood and the product of the regularization parameter and nuclear norm of the fitted regression coefficient matrix. Intended for internal use only.

Usage

```
objective(B, b, X, Y, lambda)
```

Arguments

B	fitted regression coefficient matrix
b	fitted regression intercept vector
X	covariate matrix
Y	multinomial response matrix
lambda	regularization parameter (maybe be a vector of values)

Value

a vector of objective values for the NPMR optimization problem, one for each value of `lambda`

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

References

Scott Powers, Trevor Hastie and Rob Tibshirani (2016). "Nuclear penalized multinomial regression with an application to predicting at bat outcomes in baseball." In prep.

See Also

`logL`, `nuclear`, `objectiveFast`

`objectiveFast` *Shortcut computation for NPMR objective function*

Description

Computes the objective function for NPMR more quickly than `objective` by leveraging pre-computed fitted values, which is the bottleneck in computing the objective. Intended for internal use only.

Usage

```
objectiveFast(B, P, W, lambda)
```

Arguments

B	fitted regression coefficient matrix
P	matrix of fitted multinomial class probabilities
W	vector containing indices of `P` which correspond to observed data
lambda	regularization parameter (maybe be a vector of values)

Value

a vector of objective values for the NPMR optimization problem, one for each value of `lambda`

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

References

Scott Powers, Trevor Hastie and Rob Tibshirani (2016). "Nuclear penalized multinomial regression with an application to predicting at bat outcomes in baseball." In prep.

See Also

`objective`, `nuclear`, `PGDnpmr`

`PGDnpmr` *Proximal gradient descent for nuclear penalized multinomial regression*

Description

Iterates steps of proximal gradient descent until convergence, by repeatedly taking steps in the direction of the negative of the gradient and soft-thresholding the singular values of the result. Intended for internal use only.

Usage

```
PGDnpmr(B, b, X, Y, lambda, s, group = NULL, accelerated = TRUE, eps = 1e-07,
    maxit = 1e+05, quiet = TRUE)
```

Arguments

`B`	Initial regression coefficient matrix
`b`	Initial intercept vector
`X`	Covariate matrix. May be in sparse form from `Matrix` package
`Y`	Response matrix. Each row has exactly one 1 indicating response category for that observation. All other entries are zero.
`lambda`	Vector of regularization parameter values for penalizing nuclear norm
`s`	Step size for proximal gradient descent
`group`	Vector of length equal to number of variables (ncol(X) and nrow(B)). Variables in the same group indexed by a POSITIVE integer will be penalized together (the nuclear norm of the sub-matrix of the regression coefficients will be penalized). Variables without positive integers will NOT be penalized. Default is NULL, which means there are no sub-groups; nuclear norm of entire coefficient matrix is penalized.

accelerated	Logical. Should accelerated proximal gradient descent be used? Default is TRUE.
eps	Convergence threshold. When relative change in the objective function after an interation drops below this threshold, algorithm halts.
maxit	Maximum number of iterations for proximal gradient descent.
quiet	Logical. Should output be silenced? If not, print the value of the objective function after each step of proximal gradient descent. Perhaps useful for debugging. Default is TRUE.

Value

B	Optimal value of the regression coeffient matrix at convergence
b	Optimal value of the regression intercept vector at convergence
objectivePath	Vector showing the value of the objective function at each step in proximal gradient descent
time	Time taken until convergence

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

References

Scott Powers, Trevor Hastie and Rob Tibshirani (2016). "Nuclear penalized multinomial regression with an application to predicting at bat outcomes in baseball." In prep.

See Also

npmr, prox, objective, objectiveFast

---

plot.cv.npmr *Visualize the regression coefficient matrix fit by cross-validated NPMR*

Description

Plots features (in orange) by their weights on the first two latent variables in the singular value decomposition of the regression coefficient matrix. Plots response classes (as blue arrows) by their loadings on the first two latent variables. Does this for the regression coefficient matrix fit with the value of lambda that led to the minimum cross validation error among all those tried.

Usage

```
## S3 method for class 'cv.npmr'
plot(x, feature.names = TRUE, ...)
```

Arguments

x	an object of class cv.npmr
feature.names	logical. Should the names of the covariates be used in the plot? If FALSE, use standard plotting symbol (pch=1) instead.
...	additional arguments to be passed to plot

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

References

Scott Powers, Trevor Hastie and Rob Tibshirani (2016). "Nuclear penalized multinomial regression with an application to predicting at bat outcomes in baseball." In prep.

See Also

cv.npmr, plot.npmr

Examples

```
#   Fit NPMR to simulated data

K = 5
n = 1000
m = 10000
p = 10
r = 2

# Simulated training data
set.seed(8369)
A = matrix(rnorm(p*r), p, r)
C = matrix(rnorm(K*r), K, r)
B = tcrossprod(A, C)            # low-rank coefficient matrix
X = matrix(rnorm(n*p), n, p)    # covariate matrix with iid Gaussian entries
eta = X
P = exp(eta)/rowSums(exp(eta))
Y = t(apply(P, 1, rmultinom, n = 1, size = 1))
fold = sample(rep(1:10, length = nrow(X)))

# Simulate test data
Xtest = matrix(rnorm(m*p), m, p)
etatest = Xtest
Ptest = exp(etatest)/rowSums(exp(etatest))
Ytest = t(apply(Ptest, 1, rmultinom, n = 1, size = 1))

# Fit NPMR for a sequence of lambda values without CV:
fit2 = cv.npmr(X, Y, lambda = exp(seq(7, -2)), foldid = fold)

# Produce a biplot:
plot(fit2)
```

plot.npmr *Visualize the regression coefficient matrix fit by cross-validated NPMR*

Description

Plots features (in orange) by their weights on the first two latent variables in the singular value decomposition of the regression coefficient matrix. Plots response classes (as blue arrows) by their loadings on the first two latent variables. Does this for the regression coefficient matrix fit with the value of lambda closest among all those tried to the value of `lambda` specified.

Usage

```
## S3 method for class 'npmr'
plot(x, lambda, feature.names = TRUE, ...)
```

Arguments

`x`	an object of class `npmr`
`lambda`	a single regularization parameter value
`feature.names`	logical. Should the names of the covariates be used in the plot? If FALSE, use standard plotting symbol (`pch=1`) instead.
`...`	additional arguments to be passed to `plot`

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

References

Scott Powers, Trevor Hastie and Rob Tibshirani (2016). "Nuclear penalized multinomial regression with an application to predicting at bat outcomes in baseball." In prep.

See Also

`npmr`, `plot.cv.npmr`

Examples

```
#   Fit NPMR to simulated data

K = 5
n = 1000
m = 10000
p = 10
r = 2

# Simulated training data
set.seed(8369)
```

```
A = matrix(rnorm(p*r), p, r)
C = matrix(rnorm(K*r), K, r)
B = tcrossprod(A, C)            # low-rank coefficient matrix
X = matrix(rnorm(n*p), n, p)    # covariate matrix with iid Gaussian entries
eta = X
P = exp(eta)/rowSums(exp(eta))
Y = t(apply(P, 1, rmultinom, n = 1, size = 1))

# Simulate test data
Xtest = matrix(rnorm(m*p), m, p)
etatest = Xtest
Ptest = exp(etatest)/rowSums(exp(etatest))
Ytest = t(apply(Ptest, 1, rmultinom, n = 1, size = 1))

# Fit NPMR for a sequence of lambda values without CV:
fit2 = npmr(X, Y, lambda = exp(seq(7, -2)))

# Produce a biplot:
plot(fit2, lambda = 20)
```

`predict.cv.npmr` *Make predictions from a "cv.npmr" object*

Description

Return predicted reponse class probabilities from a cross-validated NPMR model, using the value of the regularization parameter that led to the minimum cross validation error

Usage

```
## S3 method for class 'cv.npmr'
predict(object, newx, ...)
```

Arguments

`object`	an object of class `cv.npmr`
`newx`	covariate matrix on which for which to make response class probability predictions. Must have same number of columns as `X` used original to fit `object`.
`...`	ignored

Value

a matrix giving the predicted probability that each row of `newx` belongs to each class, corresponding the value of the regularization parameter that led to minimum cross validation error.

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

References

Scott Powers, Trevor Hastie and Rob Tibshirani (2016). "Nuclear penalized multinomial regression with an application to predicting at bat outcomes in baseball." In prep.

See Also

cv.npmr, predict.npmr

Examples

```
#   Fit NPMR to simulated data

K = 5
n = 1000
m = 10000
p = 10
r = 2

# Simulated training data
set.seed(8369)
A = matrix(rnorm(p*r), p, r)
C = matrix(rnorm(K*r), K, r)
B = tcrossprod(A, C)            # low-rank coefficient matrix
X = matrix(rnorm(n*p), n, p)    # covariate matrix with iid Gaussian entries
eta = X
P = exp(eta)/rowSums(exp(eta))
Y = t(apply(P, 1, rmultinom, n = 1, size = 1))
fold = sample(rep(1:10, length = nrow(X)))

# Simulate test data
Xtest = matrix(rnorm(m*p), m, p)
etatest = Xtest
Ptest = exp(etatest)/rowSums(exp(etatest))
Ytest = t(apply(Ptest, 1, rmultinom, n = 1, size = 1))

# Fit NPMR for a sequence of lambda values without CV:
fit2 = cv.npmr(X, Y, lambda = exp(seq(7, -2)), foldid = fold)

# Compute mean test error using the predict function:
-mean(log(rowSums(Ytest*predict(fit2, Xtest))))
```

`predict.npmr` *Make predictions from a "npmr" object*

Description

Return predicted reponse class probabilities from a fitted NPMR model, for each value of lambda on which the NPMR model was originally fit.

Usage

```
## S3 method for class 'npmr'
predict(object, newx, ...)
```

Arguments

object	an object of class npmr
newx	covariate matrix on which for which to make response class probability predictions. Must have same number of columns as X used original to fit object.
...	ignored

Value

a 3-dimensional array, with dimensions (nrow(newx), ncol(Y), length(lambda)). For each lambda, this array stores for that value of lambda the predicted response class probabilites for each observation.

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

References

Scott Powers, Trevor Hastie and Rob Tibshirani (2016). "Nuclear penalized multinomial regression with an application to predicting at bat outcomes in baseball." In prep.

See Also

npmr, predict.cv.npmr

Examples

```
#   Fit NPMR to simulated data

K = 5
n = 1000
m = 10000
p = 10
r = 2

# Simulated training data
set.seed(8369)
A = matrix(rnorm(p*r), p, r)
C = matrix(rnorm(K*r), K, r)
B = tcrossprod(A, C)            # low-rank coefficient matrix
X = matrix(rnorm(n*p), n, p)    # covariate matrix with iid Gaussian entries
eta = X
P = exp(eta)/rowSums(exp(eta))
Y = t(apply(P, 1, rmultinom, n = 1, size = 1))
```

```
# Simulate test data
Xtest = matrix(rnorm(m*p), m, p)
etatest = Xtest
Ptest = exp(etatest)/rowSums(exp(etatest))
Ytest = t(apply(Ptest, 1, rmultinom, n = 1, size = 1))

# Fit NPMR for a sequence of lambda values without CV:
fit2 = npmr(X, Y, lambda = exp(seq(7, -2)))

# Compute mean test error using the predict function (for each value of lambda):
getloss = function(pred, Y) {
    -mean(log(rowSums(Y*pred)))
}
apply(predict(fit2, Xtest), 3, getloss, Ytest)
```

`print.cv.npmr` *summarize a "cv.npmr" object*

Description

Print (1) the call that produced the `cv.npmr` object; (2) the value of the regularization parameter lambda that led to the minimum cross validation error; (3) the rank of the fitted regression coefficient matrix; and (4) the per-observation cross validation error using the optimal lambda.

Usage

```
## S3 method for class 'cv.npmr'
print(x, ...)
```

Arguments

`x`	an object of class `cv.npmr`
`...`	ignored

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

References

Scott Powers, Trevor Hastie and Rob Tibshirani (2016). "Nuclear penalized multinomial regression with an application to predicting at bat outcomes in baseball." In prep.

See Also

`cv.npmr`, `print.npmr`

Examples

```
#   Fit NPMR to simulated data

K = 5
n = 1000
m = 10000
p = 10
r = 2

# Simulated training data
set.seed(8369)
A = matrix(rnorm(p*r), p, r)
C = matrix(rnorm(K*r), K, r)
B = tcrossprod(A, C)            # low-rank coefficient matrix
X = matrix(rnorm(n*p), n, p)    # covariate matrix with iid Gaussian entries
eta = X
P = exp(eta)/rowSums(exp(eta))
Y = t(apply(P, 1, rmultinom, n = 1, size = 1))
fold = sample(rep(1:10, length = nrow(X)))

# Simulate test data
Xtest = matrix(rnorm(m*p), m, p)
etatest = Xtest
Ptest = exp(etatest)/rowSums(exp(etatest))
Ytest = t(apply(Ptest, 1, rmultinom, n = 1, size = 1))

# Fit NPMR for a sequence of lambda values without CV:
fit2 = cv.npmr(X, Y, lambda = exp(seq(7, -2)), foldid = fold)

# Print the NPMR fit:
fit2
```

`print.npmr` *Summarize a "npmr" object*

Description

Print the call that produced the `npmr` object and a dataframe showing, for each value of the regularization parameter on which the NPMR object was fit, the rank of the resulting regression coefficient matrix and the corresponding value of the NPMR objective function.

Usage

```
## S3 method for class 'npmr'
print(x, ...)
```

Arguments

`x`	an object of class `npmr`
`...`	ignored

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

References

Scott Powers, Trevor Hastie and Rob Tibshirani (2016). "Nuclear penalized multinomial regression with an application to predicting at bat outcomes in baseball." In prep.

See Also

npmr, print.cv.npmr

Examples

```
#   Fit NPMR to simulated data

K = 5
n = 1000
m = 10000
p = 10
r = 2

# Simulated training data
set.seed(8369)
A = matrix(rnorm(p*r), p, r)
C = matrix(rnorm(K*r), K, r)
B = tcrossprod(A, C)            # low-rank coefficient matrix
X = matrix(rnorm(n*p), n, p)    # covariate matrix with iid Gaussian entries
eta = X
P = exp(eta)/rowSums(exp(eta))
Y = t(apply(P, 1, rmultinom, n = 1, size = 1))

# Simulate test data
Xtest = matrix(rnorm(m*p), m, p)
etatest = Xtest
Ptest = exp(etatest)/rowSums(exp(etatest))
Ytest = t(apply(Ptest, 1, rmultinom, n = 1, size = 1))

# Fit NPMR for a sequence of lambda values without CV:
fit2 = npmr(X, Y, lambda = exp(seq(7, -2)))

# Print the NPMR fit:
fit2
```

---

prox *Proximal operator for nuclear norm*

Description

Return the value of the proximal operator of the nuclear norm (scaled by `threshold`) applied to a matrix

Usage

```
prox(B, threshold, group)
```

Arguments

`B`	matrix
`threshold`	scaling factor applied to the nuclear norm. In proximal gradient descent for NPMR, this is the product of the stepsize and the regularization parameter lambda
`group`	Vector of length equal to number of variables, i.e. nrow(B). Variables in the same group indexed by a POSITIVE integer will be penalized together (the nuclear norm of the sub-matrix of the regression coefficients will be penalized). Variables without positive integers will NOT be penalized. Default is NULL, which means there are no sub-groups; nuclear norm of entire coefficient matrix is penalized.

Value

the value of the proximal operator of the nuclear norm (scaled by `threshold`) applied to `B`

Author(s)

Scott Powers, Trevor Hastie, Rob Tibshirani

References

Neal Parikh and Stephen Boyd (2013) "Proximal algorithms." Foundations and Trends in Optimization 1, 3:123-231.

See Also

`nuclear`, `PGDnpmr`

`Vowel` *Vowel Recognition*

Description

Speaker independent recognition of the eleven steady state vowels of British English using a specified training set of lpc derived log area ratios.

Format

A data frame with 990 observations on the following 12 variables.

`y` Class label indicating vowel spoken

`subset` a factor with levels `test` `train`

`x.1` a numeric vector

`x.2` a numeric vector

`x.3` a numeric vector

`x.4` a numeric vector

`x.5` a numeric vector

`x.6` a numeric vector

`x.7` a numeric vector

`x.8` a numeric vector

`x.9` a numeric vector

`x.10` a numeric vector

Details

The speech signals were low pass filtered at 4.7kHz and then digitised to 12 bits with a 10kHz sampling rate. Twelfth order linear predictive analysis was carried out on six 512 sample Hamming windowed segments from the steady part of the vowel. The reflection coefficients were used to calculate 10 log area parameters, giving a 10 dimensional input space. For a general introduction to speech processing and an explanation of this technique see Rabiner and Schafer [RabinerSchafer78].

Each speaker thus yielded six frames of speech from eleven vowels. This gave 528 frames from the eight speakers used to train the networks and 462 frames from the seven speakers used to test the networks.

The eleven vowels, along with words demonstrating their sound, are: i (heed) I (hid) E (head) A (had) a: (hard) Y (hud) O (hod) C: (hoard) U (hood) u: (who'd) 3: (heard)

Source

https://archive.ics.uci.edu/ml/machine-learning-databases/undocumented/connectionist-bench/vowel/

References

D. H. Deterding, 1989, University of Cambridge, "Speaker Normalisation for Automatic Speech Recognition", submitted for PhD.

Examples

```
data(Vowel)
summary(Vowel)
```

Index