Package 'CatReg'

March 14, 2020

Type Package

Title Solution Paths for Linear and Logistic Regression Models with SCOPE Penalty

Version 1.0.0

Date 2020-03-05

Description Computes solutions for linear and logistic regression models with a nonconvex penalty (SCOPE) in an efficient path-wise fashion (Stokell, Shah and Tibshirani 2020, <arXiv:2002.12606>). The scaling of the solution paths is selected automatically. Includes functionality for selecting tuning parameter lambda by k-fold cross-validation and early termination based on information criteria. Solutions are computed by cyclical block-coordinate descent, iterating an innovative dynamic programming algorithm to compute exact solutions for each block.

License GPL (>= 2)

Imports Rcpp (>= 1.0.1),Rdpack

LinkingTo Rcpp

NeedsCompilation yes

RdMacros Rdpack

RoxygenNote 7.0.2

Author Benjamin Stokell [aut],
Daniel Grose [ctb, cre],
Rajen Shah [ctb]

Maintainer Daniel Grose <dan.grose@lancaster.ac.uk>

Repository CRAN

Date/Publication 2020-03-14 18:10:05 UTC

R topics documented:

CorrelatedDesignMatrix *Create a design matrix of categorical variables with correlated columns.*

Description

Function for use in simulations. Created design matrix of categorical variables with correlated columns

Usage

```
CorrelatedDesignMatrix(n, cov_mat, c)
```

Arguments

`n`	Number of observations
`cov_mat`	p x p covariance matrix. Controls correlations of pairs of marginally U[0,1] random variables that are subsequently binned to assign categories for each variable
`c`	Number of categories within each variable

Value

A data frame of categorical (factor) variables.

Examples

```
# Generate matrix of marginal U[0,1] variables, 0.5 pairwise correlation, that are
# discretised into factor variables
cov_mat = 0.5 * diag(10) + 0.5 * matrix(1, 10, 10)
x = CorrelatedDesignMatrix(100, cov_mat, 8)
```

predict.scope *Computes SCOPE predictions*

Description

Computes SCOPE predictions on new data.

Usage

```
## S3 method for class 'scope'
predict(object, newdata, interceptxlinear = FALSE, ...)
```

Arguments

`object`	SCOPE model as outputted by scope. Must have simply.the.best = TRUE
`newdata`	New covariates on which to make predictions. Must be of the same form as the model was trained on
`interceptxlinear`	If intercept is to be included in the model and is not in the column span of the continuous variables in x, set to FALSE (default). Must match format of training data.
`...`	Additional arguments to pass to other `predict` methods

Value

Returns n-length vector of predictions

See Also

`scope`

`predict.scope.logistic` *Computes SCOPE logistic predictions*

Description

Computes SCOPE logistic predictions on new data

Usage

```
## S3 method for class 'scope.logistic'
predict(object, newdata, probs = TRUE, interceptxlinear = FALSE, ...)
```

Arguments

`object`	SCOPE model as outputted by scope.logistic. Must have simply.the.best = TRUE
`newdata`	New covariates on which to make predictions. Must be of the same form as the model was trained on
`probs`	If TRUE returns probabilities, if FALSE returns binary predictions
`interceptxlinear`	If intercept is to be included in the model and is not in the column span of the continuous variables in x, set to FALSE (default). Must match format of training data.
`...`	Additional arguments to pass to other `predict` methods

Value

Returns n-length vector of predictions

See Also

`scope.logistic`

`scope` *Compute solution for SCOPE linear models.*

Description

Computes solution for SCOPE linear models. Performs K-fold cross-validation for regularisation parameter lambda and can incorporate both linear and categorical (including logical) variables. See Stokell, Shah and Tibshirani (2020).

Usage

```
scope(
  x,
  y,
  gamma = 8,
  sd = NULL,
  AIC = TRUE,
  mBICconstant = 25,
  default.lambdaseq = TRUE,
  default.length = 100,
  lambda.min.ratio = 0.01,
  lambdaseq = NULL,
  TerminateEpsilon = 1e-07,
  interceptxlinear = FALSE,
  max.iter = 1000,
  BICearlystop = TRUE,
  BICterminate = 20,
  silent = TRUE,
  K = 5,
  return.full.beta = FALSE,
  simply.cross.validated = FALSE,
  grid.safe = 10,
  blockorder = NULL,
  FoldAssignment = NULL
)
```

Arguments

`x`	Data frame of covariates. Can include a mix of continuous and categorical covariates (no scaling of continuous covariates is performed within the program). By default an intercept will be added; see interceptxlinear
`y`	Response vector of length n
`gamma`	Concavity parameter in MCP; see Zhang (2010) Nearly unbiased estimation with minimax concave penalty
`sd`	Standard deviation of noise used for calibration of parameter lambda. This is recommended to be left alone
`AIC`	Controls whether information criterion for early stopping is AIC (=TRUE) or mBIC (=FALSE)
`mBICconstant`	If using mBIC, this is the parameter m
`default.lambdaseq`	If using automatically generated sequence of lambda values set to TRUE. Do not set to FALSE without good reason
`default.length`	Length of sequence of automatically generated lambda values
`lambda.min.ratio`	Ratio of largest to smallest value on sequence of lambda values
`lambdaseq`	If default.lambdaseq = FALSE then add path of lambda values here
`TerminateEpsilon`	Epsilon for convergence criterion, is multiplied by null deviance to get terminate criterion for objective value
`interceptxlinear`	If intercept is to be included in the model and is not in the column span of the continuous variables in x, set to FALSE (default).
`max.iter`	Maximum number of iterations at each value of lambda
`BICearlystop`	If information criterion is to be used to stop computation early, set to TRUE
`BICterminate`	Specifies how many values of lambda to be computed after the minimum value of the information criterion is reached
`silent`	If FALSE then progress updates will be printed as solutions are computed. Useful for tuning and diagnosing convergence issues.
`K`	Number of folds in cross-validation. If K = 1, no cross-validation is performed
`return.full.beta`	If TRUE then beta.full will be returned, else just the cross-validation optimal beta.best will be returned
`simply.cross.validated`	If TRUE then cross-validation scores for each value of lambda will be returned, but not the estimates themselves
`grid.safe`	As the automatically generated sequence of lambda values is adjusted during the first fold but fixed thereafter. For subsequent folds, this sets computation to begin at a larger value of lambda to ensure that the first solution along the path is zero so as to maintain the advantages of the pathwise approach. This specifies how many larger values of lambda should be used

`blockorder`	By default the order in block coordinate descent is randomly sampled. Alternatively a permutation vector can be included here
`FoldAssignment`	By default the assignments for cross-validation are randomly sampled automatically. Alternatively assignments can be included here

Value

A list of objects. Some may not be returned depending on value of arguments K, simply.cross.validated, return.full.beta.

- lambdaseq - A matrix of the values of lambda used to compute the solution path. Columns correspond to different points on the path, rows correspond to the categorical variables. Lambda is scaled depending on the number of categories present in the data.
- cverrors - Provided K > 1 then the cross-validation error for each point on the grid will be returned
- beta.full - Contains full solution path. If K > 1 then will only be returned if simply.cross.validated = FALSE and return.full.beta = TRUE. First object [[1]] is coefficients of continuous variables, [[2]] is a list of coefficients for categorical variables
- beta.best - Contains solution at CV-optimal point. Requires K > 1 to be returned. This must not be NULL in order to use predict.scope. First object [[1]] is coefficients of continuous variables, [[2]] is a list of coefficients for categorical variables
- fold.assign - Contains fold assignments for cross-validation

References

Stokell BG, Shah RD, Tibshirani RJ (2020). “Modelling High-Dimensional Categorical Data Using Nonconvex Fusion Penalties.” 2002.12606.

Examples

```
set.seed(1)
x = UniformDesignMatrix(100, 5, 8)
y = (x[ ,1 ] == "A1") + (x[ ,1 ] == "B1") +
    (x[ ,1 ] == "C1") + (x[ ,1 ] == "D1") +
    (x[ ,2 ] == "A2") + (x[ ,2 ] == "B2") +
    (x[ ,2 ] == "C2") + (x[ ,2 ] == "D2") + rnorm(100)
scope_mod = scope(x, y)
x_new = UniformDesignMatrix (10, 5, 8)
predict(scope_mod, x_new)
```

scope.logistic *Computes solution for SCOPE logistic models*

Description

Computes solution for SCOPE logistic models, computing along a path and iterating local quadratic approximations at each point. Performs K-fold cross-validation for regularisation parameter lambda and can incorporate both linear and categorical (including logical) variables. This function uses a Proximal Newton algorithm and is not guaranteed to converge. It is recommended for developer use only. See Stokell, Shah and Tibshirani (2020).

Usage

```
scope.logistic(
  x,
  y,
  gamma = 100,
  sd = NULL,
  AIC = TRUE,
  mBICconstant = 5,
  default.lambdaseq = TRUE,
  default.length = 160,
  lambda.min.ratio = 0.005,
  lambdaseq = NULL,
  TerminateEpsilon = 1e-07,
  interceptxlinear = FALSE,
  max.iter = 1000,
  max.out.iter = 1000,
  BICearlystop = TRUE,
  BICterminate = 25,
  silent = TRUE,
  K = 5,
  return.full.beta = FALSE,
  simply.cross.validated = FALSE,
  grid.safe = 10,
  blockorder = NULL,
  FoldAssignment = NULL,
  zero.penalty = FALSE
)
```

Arguments

x	Data frame of covariates. Can include a mix of continuous and categorical covariates (no scaling of continuous covariates is performed within the program). By default an intercept will be added; see interceptxlinear
y	Response vector of length n

`gamma`	Concavity parameter in MCP; see Zhang (2010) Nearly unbiased estimation with minimax concave penalty
`sd`	Standard deviation of noise used for calibration of parameter lambda. This is recommended to be left alone
`AIC`	Controls whether information criterion for early stopping is AIC (=TRUE) or mBIC (=FALSE)
`mBICconstant`	If using mBIC, this is the parameter m
`default.lambdaseq`	If using automatically generated sequence of lambda values set to TRUE. Do not set to FALSE without good reason
`default.length`	Length of sequence of automatically generated lambda values
`lambda.min.ratio`	Ratio of largest to smallest value on sequence of lambda values
`lambdaseq`	If default.lambdaseq = FALSE then add path of lambda values here
`TerminateEpsilon`	Epsilon for convergence criterion, is multiplied by null deviance to get terminate criterion for objective value
`interceptxlinear`	If intercept is to be included in the model and is not in the column span of the continuous variables in x, set to FALSE (default).
`max.iter`	Maximum number of iterations at each local quadratic approximation
`max.out.iter`	Maximum number of local quadratic approximations at each value of lambda
`BICearlystop`	If information criterion is to be used to stop computation early, set to TRUE
`BICterminate`	Specifies how many values of lambda to be computed after the minimum value of the information criterion is reached
`silent`	If FALSE then progress updates will be printed as solutions are computed
`K`	Number of folds in cross-validation. If K = 1, no cross-validation is performed
`return.full.beta`	If TRUE then beta.full will be returned, else just the cross-validation optimal beta.best will be returned
`simply.cross.validated`	If TRUE then cross-validation scores for each value of lambda will be returned, but not the estimates themselves
`grid.safe`	As the automatically generated sequence of lambda values is adjusted during the first fold but fixed thereafter. For subsequent folds, this sets computation to begin at a larger value of lambda to ensure that the first solution along the path is zero so as to maintain the advantages of the pathwise approach. This specifies how many larger values of lambda should be used
`blockorder`	By default the order in block coordinate descent is randomly sampled. Alternatively a permutation vector can be included here
`FoldAssignment`	By default the assignments for cross-validation are randomly sampled automatically. Alternatively assignments can be included here
`zero.penalty`	Fits logistic regression model with zero penalty. Primarily used for debugging, do not set to TRUE

Value

A list of objects. Some may not be returned depending on value of arguments K, simply.cross.validated, return.full.beta.

- lambdaseq - A matrix of the values of lambda used to compute the solution path. Columns correspond to different points on the path, rows correspond to the categorical variables. Lambda is scaled depending on the number of categories present in the data.
- cverrors - Provided K > 1 then the cross-validation error for each point on the grid will be returned
- beta.full - Contains full solution path. If K > 1 then will only be returned if simply.cross.validated = FALSE and return.full.beta = TRUE. First object [[1]] is coefficients of continuous variables, [[2]] is a list of coefficients for categorical variables
- beta.best - Contains solution at CV-optimal point. Requires K > 1 to be returned. This must not be NULL in order to use predict.scope. First object [[1]] is coefficients of continuous variables, [[2]] is a list of coefficients for categorical variables
- fold.assign - Contains fold assignments for cross-validation

References

Stokell BG, Shah RD, Tibshirani RJ (2020). "Modelling High-Dimensional Categorical Data Using Nonconvex Fusion Penalties." 2002.12606.

Examples

```
## Not run:
x = UniformDesignMatrix(200, 5, 5)
y = (as.integer(x[ , 1 ]) < 3) + rnorm(200)
y = as.integer(y > 0.8)
scope_mod = scope.logistic(x, y)
x_new = UniformDesignMatrix(10, 5, 5)
predict(scope_mod, x_new)

## End(Not run)
```

`UniformDesignMatrix` *Create a design matrix of categorical variables.*

Description

Function for use in simulations, creating design matrix of categorical variables where each column is uniformly randomly distributed and independent.

Usage

```
UniformDesignMatrix(n, p, c)
```

Arguments

n	Number of observations
p	Number of variables
c	Number of categories within each variable

Value

A data frame of categorical (factor) variables.

Examples

```
x = UniformDesignMatrix(100, 10, 8)
```

Index