Package ‘predkmeans’

December 17, 2019

Title  Covariate Adaptive Clustering
Version 0.1.1
Author Joshua Keller
Maintainer Joshua Keller <joshua.keller@colostate.edu>
Description Implements the predictive k-means method for clustering observations, using a mixture of experts model to allow covariates to influence cluster centers. Motivated by air pollution epidemiology settings, where cluster membership needs to be predicted across space. Includes functions for predicting cluster membership using spatial splines and principal component analysis (PCA) scores using either multinomial logistic regression or support vector machines (SVMs). For method details see Keller et al. (2017) <doi:10.1214/16-AOAS992>.

Imports Rcpp (>= 0.11.5), maxLik, e1071, mgcv
Suggests
License GPL-3 | file LICENSE
LinkingTo Rcpp, RcppArmadillo
LazyData true
RoxygenNote 7.0.2
NeedsCompilation yes
Repository CRAN
Date/Publication 2019-12-16 23:50:02 UTC

R topics documented:

predkmeans-package ................................................. 2
assignCluster ......................................................... 2
createCVgroups ....................................................... 3
createPCAmodelmatrix ............................................... 4
createTPRSSmodelmatrix ............................................ 5
mlogit ................................................................. 6
predictionMetrics ..................................................... 8
predictML.predkmeans ............................................... 9
predkmeans .......................................................... 12
predkmeansCVest .................................................... 14
relevel.predkmeans ................................................ 16
assignCluster

Description

Clusters multivariate exposures, using a mixture of experts model to allow covariates to influence cluster centers. Motivated by air pollution epidemiology settings, where cluster membership needs to be predicted across space. Includes functions for predicting cluster membership using spatial splines and principal component analysis (PCA) scores using either multinomial logistic regression or support vector machines (SVMs). For method details see Keller et al. (2017) <doi:10.1214/16-AOAS992>

assignCluster

Make Cluster Assignments

Description

Assigns observation to the nearest cluster center, using squared Euclidean distance.

Usage

assignCluster(X, centers)

Arguments

X       matrix of observations
centers  matrix of cluster centers

Value

A vector of cluster labels

Author(s)

Joshua Keller

Examples

X <- matrix(rnorm(100*5), nrow=100, ncol=5)
centers <- matrix(runif(3*5), nrow=3, ncol=5)

cl <- assignCluster(X, centers)
table(cl)
**createCVgroups**

**Creating k-fold Cross-Validation Groups**

**Description**

Splits a vector of observation names or indices into a list of k groups, to be used as cross-validation (CV) test groups.

**Usage**

```r
createCVgroups(x = NULL, n = length(x), k = 10, useNames = TRUE)
```

**Arguments**

- `x`: vector of observation ID’s (character or numeric) to split into cv groups.
- `n`: number of observations to split into cv groups. Defaults to the length of `x`, but can also be provided instead of `x`.
- `k`: number of cross-validation groups. Must be less than or equal to `n`.
- `useNames`: logical indicator of whether the names of `x` should be used to identify observations within cv groups.

**Value**

A list of length `k` giving the IDs of observations within each test group.

**Author(s)**

Joshua Keller

**See Also**

`predkmeansCVest` `predkmeansCVpred`

**Examples**

```r
# 5-fold groups
cv5 <- createCVgroups(n=100, k=5)
cv5

# Leave-one-out
cvLOO <- createCVgroups(n=100, k=0)
cvLOO
```
createPCAmatrix 

Create Principal Component Analysis (PCA) scores matrix

Description

Wrapper function for creating PCA scores to be used in a regression analysis.

Usage

createPCAmatrix(
  data,
  ncomps,
  covarnames = colnames(data),
  center = TRUE,
  scale = TRUE,
  matrixonly = TRUE
)

Arguments

  data          Matrix or data frame of data
  ncomps        Number of PCA components to return.
  covarnames    Names of variables or column numbers in data on which the PCA is to be run.
  center        Logical indicator of whether data should be centered. Passed to prcomp.
  scale         Logical indicator of whether data should be scaled. Passed to prcomp.
  matrixonly    Logical indicator of whether only the model matrix should be returned, or the full output from prcomp.

Details

This is a wrapper around prcomp, which does the necessary computation.

Value

If matrixonly=TRUE, a matrix of PCA scores. Otherwise a list containing two elements: X, a matrix of scores, and pca, the output from prcomp.

Author(s)

Joshua Keller

See Also

createTPRSmodelmatrix, predkmeansCVest
Examples

```r
n <- 100
d <- 15
X <- matrix(rnorm(n*d), ncol=d, nrow=n)
X <- as.data.frame(X)
x <- createPCAmodelmatrix(data=X, ncomps=2)
```

createTPRSMmodelmatrix  
Create matrix of Thin-Plate Regression Splines (TPRS)

Description

Wrapper function for creating a matrix of thin-plate regression splines (TPRS) to be used in a regression analysis.

Usage

```r
createTPRSMmodelmatrix(
  data,
  df = 5,
  covarnames = NULL,
  xname = "x",
  yname = "y",
  TPRSfx = TRUE,
  matrixonly = TRUE
)
```

Arguments

data  Matrix or data frame of data
df  Degrees of freedom for thinplate splines. This does not include an intercept, so the k argument of s() is k = df + 1.
covarnames  Names of other covariates to be included in the model matrix.
xname  Name of variable the provides the x-coordinate of location.
yname  Name of variable the provides the y-coordinate of location.
TPRSfx  Should the TPRS degrees of freedom be fixed. Passed as the fx argument to s().
matrixonly  Logical indicator of whether only the model matrix should be returned, or the full output from gam.

Author(s)

Joshua Keller
See Also

createPCAmodelmatrix, predkmeansCVest

Examples

n <- 200
x <- runif(n=n, 0, 100)
y <- runif(n=n, 0, 100)
d <- data.frame(x=x, y=y)
mx <- createTPRSMmodelmatrix(data=d, df=5)

mlogit

Multinomial Logistic Regression

Description

Solves a multinomial logistic problem using Newton-Raphson method

Usage

mlogit(
  Y, 
  X, 
  beta = NULL, 
  add.intercept = FALSE, 
  betaOnly = FALSE, 
  tol.zero = 1e-08, 
  verbose = T, 
  suppressFittedWarning = FALSE, 
  maxNR.print.level = 0, 
  iterlim = 150, 
  checkY = TRUE
)

Arguments

Y

A matrix of the outcomes, with K columns for the K groups. Row sums of the
matrix should be equal to one, but entries do not have to be 0/1 (but they should
be positive), i.e. this is a matrix of hard or soft assignments to K categories. The
first column is used as the reference category.

X

matrix of covariates for regression. Should have the same number of rows (ob-
servations) as Y. Coefficients for all parameters in X are computed for K-1
groups. The coefficients corresponding to the first column of Y are assumed
to be zero.

beta

starting values for the optimziation. Should be given as a matrix of column
vectors, each vector a different starting value. If null, defaults to zeros.
add.intercept  a logical indicator of whether an intercept column should be added to X
betaOnly      logical indicator of whether only the parameter estimates beta should be returned. Otherwise, beta is returned along with fitted objects. See Output.
tol.zero      the tolerance threshold for considering a fitted value as equal to zero. Used for warning about fitted values of 0/1. Is NOT part of the optimization control parameters.
verbose      logical indicator that controls whether text indicating progress is output to display
suppressFittedWarning  indicator of whether or not warnings about fitted values of 1 are returned
maxNR.print.level numeric value giving the level of output produced by maxNR. see ?maxNR for details. Defaults to 0.
iterlim      iteration limit for maxNR. Defaults to 150.
checkY       indicator for whether Y should be checked to be a valid assignment matrix. Set to FALSE if using decimal values in Y.

Details

The optimization is done using the maxNR function from the maxLik package. The log-likelihood function, along with its gradient and hessian, are implemented as C++ functions (via the RcppArmadillo package).

Value

A list containing the following:

beta  a p x K matrix of parameter estimates corresponding to the K columns of Y and p covariates in X
fitted01  indicator of whether fitted values of 1 were present.
fitted  the fitted probabilities
res.best  the best result from the maxNR fit
status  small data frame summarizing the status of the fits
res.all  a list containing the results from all maxNR fits

Author(s)

Joshua Keller

See Also

predkmeans
Examples

```r
n <- 2000
X <- cbind(1,
    matrix(rnorm(2*n), nrow=n, ncol=2),
    rbinom(n, size=1, prob=0.3))
beta <- cbind(rep(0, 4),
    c(0.5, 1, 0, -1),
    c(0, 2, 2, 0))
probs <- exp(X %*% beta)
probs <- probs/rowSums(probs)
Y <- t(apply(probs, 1, function(p) rmultinom(1, 1, p)))
mfit <- mlogit(Y=Y, X=X, betaOnly=TRUE)
mfit
```

**predictionMetrics**  
*Measures of Prediction Performance*

**Description**

Computes several measures of performance for cluster label prediction.

**Usage**

```r
predictionMetrics(centers, cluster.pred, X, labels = TRUE)
```

**Arguments**

- `centers`: Matrix of Cluster centers
- `cluster.pred`: Vector of predicted cluster membership. Should be integers or names corresponding to rows of `centers`.
- `X`: Matrix of observations at prediction locations.
- `labels`: Logical indicating whether cluster prediction and

**Value**

A list with the following elements:

- `MSPE`: Mean squared prediction error. Sum of squared distances between observations and predicted cluster centers.
- `wSS`: Within-cluster sum-of-squares. Sum of squared distances between observations at prediction locations and best (i.e. closest) cluster center.
- `MSME`: Mean squared misclassification error. Sum of squared distances between predicted cluster center and best (i.e. closest) cluster center.
- `pred.acc`: Proportion of cluster labels correctly predicted.
- `cluster.pred`: Predicted cluster assignments (same as argument provided).
- `cluster.assign`: Integer vector of 'best' cluster assignments (i.e. assignment to closest cluster center)
predictML.predkmeans

Author(s)
Joshua Keller

References

See Also
predictML

Examples
```
n <- 100
d <- 5  # Dimension of exposure
K <- 3  # Number of clusters
X <- matrix(rnorm(n*d), ncol=d, nrow=n)
centers <- matrix(runif(d*K), nrow=K, ncol=d)
cluster_pred <- sample(1:K, size=n, replace=TRUE)
metrics <- predictionMetrics(centers, cluster.pred=cluster_pred, X=X)
metrics[c("MSPE", "wSS", "MSME", "pred.acc")]
```

Description

Predicts cluster membership using either multinomial logistic regression or SVMs.

Usage
```
## S3 method for class 'predkmeans'
predictML(
  object = NULL,
  centers = object$centers,
  K = nrow(centers),
  R,
  Rstar,
  Xstar = NULL,
  tr.assign = object$cluster,
  muStart = "random",
  maxitMlogit = 500,
  verbose = 1,
  nMlogitStarts = 1,
  mlogit.control = list(suppressFittedWarning = TRUE),
  ...
)`
predictML::predkmeans

## S3 method for class 'predkmeans'
predictSVM(
  object = NULL,
  centers = object$centers,
  R,
  Rstar,
  K = nrow(centers),
  Xstar = NULL,
  tr.assign = object$cluster,
  svm.control = list(gamma = c(1/(2:1), 2), cost = seq(20, 100, by = 20)),
  ...
)

## S3 method for class 'predkmeans'
predictMixExp(object, R, Rstar = NULL, ...)

### Arguments

- **object**: A predkmeans object, from which the cluster centers will be extracted.
- **centers**: Matrix of cluster centers, assumed to be K-by-p
- **K**: Number of clusters
- **R**: matrix of covariates for observations to be predicted at.
- **Rstar**: matrix of covariates at training locations
- **Xstar**: matrix of observation at training locations. Either this or `tr.assign` is required.
- **tr.assign**: vector of cluster assignments at training locations. By default, extracted from `object`.
- **muStart**: starting value for cluster centers in mlogit optimization (IDEA: change to pull from predkmeans object?). If not provided, starting values are selected randomly.
- **maxitMlogit**: Maximum number of iterations for mlogit in prediction
- **verbose**: integer indicating amount of output to be displayed
- **nMlogitStarts**: number of mlogit starts to use in estimation of parameters
- **mlogit.control**: list of control parameters to be passes to mlogit
- **...**: Unused additional arguments
- **svm.control**: list of options for best.svm

### Details

Function for predicting cluster membership in clusters identified by k-means or predictive k-means using multinomial logistic regression or support vector machines (SVMs). For multinomial logistic regression, parameter estimation is handled by mlogit. The SVMs are fit using best.svm from e1071 package.
Because this prediction includes return information about cluster assignment and prediction model parameters, this method is deliberately distinct from the generic predict functions.

The predictMixExp function provides predictions from the 'working' cluster assignments created as part of the mixture of experts algorithm from predkmeans.

Value

A list containing some or all of the following elements:

- tr.assign: Cluster assignments at training locations
- mlfit: A subset of the mlogit object returned by the function of that name
- beta: Estimated model parameters
- test.pred: Predicted cluster assignments at test locations

Author(s)

Joshua Keller

See Also

mlogit, predkmeans, predictionMetrics

Other methods for predkmeans objects: relevel.predkmeans()

Examples

```r
n <- 200
r1 <- rnorm(n)
r2 <- rnorm(n)
u1 <- rbinom(n, size=1, prob=0)
ccluster <- ifelse(r1<0, ifelse(u1, "A", "B"), ifelse(r2<0, "C", "D"))
mu1 <- c(A=2, B=2, C=-2, D=-2)
mu2 <- c(A=1, B=-1, C=-1, D=-1)
x1 <- rnorm(n, mu1[cluster], 4)
x2 <- rnorm(n, mu2[cluster], 4)
R <- cbind(1, r1, r2)
X <- cbind(x1, x2)
pkm <- predkmeans(X=cbind(x1, x2), R=R, K=4)
n_pred <- 50
Rnew <- cbind(1, r1=rnorm(n_pred), r2=rnorm(n_pred))
pkmPred <- predictML(pkm, R=Rnew, Rstar=R)
pkmPred$test.pred
```
predictive K-means Clustering

Description

Uses a Mixture-of-experts algorithm to find cluster centers that are influenced by prediction covariates.

Usage

predkmeans(
  X,
  R,
  K,
  mu = NULL,
  muStart = c("kmeans", "random"),
  sigma2 = 0,
  sigma2fixed = FALSE,
  maxitEM = 100,
  tol = 1e-05,
  convEM = c("both", "mu", "gamma"),
  nStarts = 1,
  maxitMlogit = 500,
  verbose = 0,
  muRestart = 1000,
  returnAll = FALSE,
  ...
)

Arguments

X An n by p matrix or data frame of data to be clustered.
R Covariates used for clustering. Required unless doing k-means clustering (i.e. sigma2=0 and sigma2fixed=TRUE).
K Number of clusters
mu starting values for cluster centers. If NULL (default), then value is chosen according to muStart.
muStart Character string indicating how initial value of mu should be selected. Only used if mu=NULL. Possible values are "random" or "kmeans" (default).
sigma2 starting value of sigma2. If set to 0 and sigma2fixed=TRUE, the standard k-means is done instead of predictive k-means.
sigma2fixed Logical indicating whether sigma2 should be held fixed. If FALSE, then sigma2 is estimated using Maximum Likelihood.
maxitEM Maximum number of EM iterations for finding the Mixture of Experts solution. If doing regular k-means, this is passed as iter.max.
predkmeans

tol convergence criterion
convEM controls the measure of convergence for the EM algorithm. Should be one of "mu", "gamma", or "both". Defaults to "both." The EM algorithm stops when the Frobenius norm of the change in mu, the change in gamma, or the change in mu and the change in gamma is less than 'tol'.
nStarts number of times to perform EM algorithm
maxitMlogit Maximum number of iterations in the mlogit optimization (nested within EM algorithm)
verbose numeric vector indicating how much output to produce
muRestart Gives max number of attempts at picking starting values. Only used when muStart='random'. If selected starting values for mu are constant within each cluster, then the starting values are re-selected up to muRestart times.
returnAll A list containing all nStarts solutions is included in the output.

Additional arguments passed to mlogit

Details

A thorough description of this method is provided in Keller et al. (2017). The algorithm for solving the mixture of Experts model is based upon the approach presented by Jordan and Jacobs (1994). If sigma2 is 0 and sigm2fixed is TRUE, then standard k-means clustering (using kmeans) is done instead.

Value

An object of class predkmeans, containing the following elements:

res.best A list containing the results from the best-fitting solution to the Mixture of Experts problem:

mu Maximum-likelihood estimate of intercepts from normal mixture model. These are the cluster centers.
gamma Maximum-likelihood estimates of the mixture coefficients.
sigma2 If sigma2fixed=FALSE, the maximum likelihood estimate of sigma2
conv Indicator of convergence.
objective Value of the log-likelihood.
iter Number of iterations.
mfit A subset of output from mlogit.
center Matrix of cluster centers
cluster Vector of cluster labels assigned to observations
K Number of clusters
sigma2 Final value of sigma^2.
wSS Mean within-cluster sum-of-squares
sigma2fixed Logical indicator of whether sigma2 was held fixed
predkmeansCVest

Author(s)
Joshua Keller

References


See Also
**predictML.predkmeans**, **predkmeansCVest**

Examples

```r
n <- 200
r1 <- rnorm(n)
r2 <- rnorm(n)
u1 <- rbinom(n, size=1,prob=0)
cluster <- ifelse(r1<0, ifelse(u1, "A", "B"), ifelse(r2<0, "C", "D"))
mu1 <- c(A=2, B=2, C=-2, D=-2)
mu2 <- c(A=1, B=-1, C=-1, D=-1)
x1 <- rnorm(n, mu1[cluster], 4)
x2 <- rnorm(n, mu2[cluster], 4)
R <- model.matrix(~r1 + r2)
X <- cbind(x1, x2)
pkm <- predkmeans(X=cbind(x1, x2), R=R, K=4)
summary(pkm)
```

predkmeansCVest

**Cross-validation of Predictive K-means Clustering**

Description

Performs cross-validation of predictive k-means clustering and cluster prediction.

Usage

```r
predkmeansCVest(
  X,
  R,
  K,
  cv.groups = 10,
  sigma2 = 0,
  sigma2fixed = FALSE,
  scale = TRUE,
)```
covarnames = colnames(R),
PCA = FALSE,
PCAcontrol = list(covarnames = colnames(R), ncomps = 5),
TPRS = FALSE,
TPRScontrol = list(df = 5, xname = "x", yname = "y"),
returnAll = FALSE,
...)

predkmeansCVpred(
  object,
  X = object$X,
  R = object$R,
  method = c("ML", "MixExp", "SVM"),
  ...
)

Arguments

X  Outcome data
R  Covariates. Coerced to data frame.
K  Number of clusters
cv.groups  A list providing the cross-validation groups for splitting the data. Groups for splitting the data. Alternatively, a single number giving the number of groups into which the data are randomly split. A value of '0' implies leave-one-out. Defaults to 10.
sigma2  Starting value of sigma2. Setting sigma2=0 and sigma2fixed=TRUE results in regular k-means clustering.
sigma2fixed  Logical indicating whether sigma2 should be held fixed. If FALSE, then sigma2 is estimated using Maximum Likelihood.
scale  Should the outcomes be re-scaled within each training group?
covarnames  Names of covariates to be included directly.
PCA  Logical indicator for whether PCA components should be computed from R.
PCAcontrol  Arguments passed to createPCAmatrix. This includes ncomps.
TPRS  Logical indicator for whether thin-plate regression splines should be created and added to covariates.
TPRScontrol  Arguments passed to createTPRSmodelmatrix. This includes df.
returnAll  A list containing all nStarts solutions is included in the output.
...  Additional arguments passed to either predkmeansC or the prediction method.
object  A predkmeansCVest object.
method  Character string indicating which prediction method should be used. Options are ML, MixExp, and SVM. See predictML for more information.
Details
These wrappers are designed to simplify cross-validation of a dataset. For models including thin-plate regression splines (TPRS) or principal component analysis (PCA) scores, these functions will re-evaluate the TPRS basis or PCA decomposition on each training set.

Author(s)
Joshua Keller

See Also
predkmeans, createPCAmodelmatrix, createTPRSmodelmatrix

Examples
```r
n <- 200
r1 <- rnorm(n)
r2 <- rnorm(n)
u1 <- rbinom(n, size=1, prob=0)
class <- ifelse(r1<0, ifelse(u1, "A", "B"), ifelse(r2<0, "C", "D"))
mu1 <- c(A=2, B=2, C=-2, D=-2)
mu2 <- c(A=1, B=-1, C=-1, D=-1)
x1 <- rnorm(n, mu1[class], 4)
x2 <- rnorm(n, mu2[class], 4)
R <- model.matrix(~r1 + r2)
X <- cbind(x1, x2)
pkmcv <- predkmeansCVest(X=cbind(x1, x2),
                         R=R, K=4, nStarts=4, cv.groups = 5,
                         TPRS=FALSE, PCA=FALSE, covarnames=colnames(R))
```

```r
relevel.prdkmens  Re-order cluster labels
```

Description
Function for re-ordering the order of clusters in a predkmeans object.

Usage
```r
## S3 method for class 'predkmeans'
relevel(x, ref = NULL, order = NULL, ...)
```

Arguments
- `x` object of class predkmeans
- `ref` New reference group ("Cluster 1"). Only used if order is NULL.
- `order` New order of clusters.
- `...` Ignored additional arguments.
Details

The elements of the order argument should refer to the current position of clusters, with the position giving the new order. So \( c(3, 1, 2) \) moves 1 to 2, 2 to 3, and 3 to 1.

Author(s)

Joshua Keller

See Also

Other methods for predkmeans objects: \texttt{predictML.predkmeans()}

Examples

```r
n <- 200
r1 <- rnorm(n)
r2 <- rnorm(n)
u1 <- rbinom(n, size=1, prob=0)
cluster <- ifelse(r1<0, ifelse(u1, "A", "B"), ifelse(r2<0, "C", "D"))
mu1 <- c(A=2, B=2, C=-2, D=-2)
mu2 <- c(A=1, B=-1, C=-1, D=-1)
x1 <- rnorm(n, mu1[cluster], 4)
x2 <- rnorm(n, mu2[cluster], 4)
R <- model.matrix(~r1 + r2)
X <- cbind(x1, x2)
pkm <- predkmeans(X=cbind(x1, x2), R=R, K=4)
table(pkm$cluster)

# Move cluster '4' to be first
pkm2 <- relevel(pkm, ref=4)
table(pkm2$cluster)

# Re-order based upon number of observations in each cluster
pkm3 <- relevel(pkm, order=order(table(pkm$cluster), decreasing=TRUE))
table(pkm3$cluster)
```
assignCluster, 2
createCVgroups, 3
createPCAmodelmatrix, 4, 6, 15, 16
createTPRSmodelmatrix, 4, 5, 15, 16
gam, 5
kmeans, 13
maxNR, 7
mlogit, 6, 11, 13
prcomp, 4
predictionMetrics, 8, 11
predictMixExp (predictML.predkmeans), 9
predictML, 9, 15
predictML (predictML.predkmeans), 9
predictML.predkmeans, 9, 14, 17
predictSVM (predictML.predkmeans), 9
predkmeans, 7, 11, 12, 15, 16
predkmeans-package, 2
predkmeansCVest, 4, 6, 14, 14
predkmeansCVpred (predkmeansCVest), 14
relevel.predkmeans, 11, 16