Package ‘cpfa’

June 20, 2022

Type Package

Title Classification with Parallel Factor Analysis

Version 1.0-4

Date 2022-06-20

Author Matthew A. Snodgress <snodg031@umn.edu>

Maintainer Matthew A. Snodgress <snodg031@umn.edu>

Depends multiway, glmnet, e1071, randomForest, nnet

Imports foreach, doParallel

Description Classification using Richard A. Harshman's Parallel Factor Analysis (Parafac) model-1 fit to a three-way or four-way data tensor/array. See Harshman and Lundy (1994): <doi:10.1016/0167-9473(94)90132-5>. Uses Parafac factor weights from one mode of this model as predictors to tune parameters for one or more classification methods via a k-fold cross-validation procedure. Supports penalized logistic regression, support vector machine, random forest, and feed-forward neural network. Supports binary and multiclass classification. Predicts class labels or class probabilities and calculates multiple classification performance measures. Parallel computing is implemented via the 'parallel' and 'doParallel' packages.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2022-06-20 14:20:06 UTC

R topics documented:

`cpfa` *Tuning for Classification with Parallel Factor Analysis*

Description

Fits Richard A. Harshman's Parallel Factor Analysis (Parafac) model-1 to a three-way or four-way data tensor/array. Uses Parafac factor weights from a single mode of this model as predictors to tune parameters for one or more classification methods via a k-fold cross-validation procedure (see package vignette for procedure's details). Supports binary and multiclass classification.

Usage

```
cpfa(x, y, nfac = 1, nfolds = 10, foldid = NULL, prior = NULL,
           method = c("PLR", "SVM", "RF", "NN"),
           family = c("binomial", "multinomial"),
           alpha = NULL, lambda = NULL, cost = NULL, gamma = NULL,
           ntree = NULL, nodesize = NULL, size = NULL, decay = NULL,
           parallel = FALSE, cl = NULL,
           verbose = TRUE, cmode = NULL, ...)
```

Arguments

`x`	Three-way or four-way data array. See note below.
`y`	A factor with two or more levels containing class labels. For binary case, ensure the order of factor levels (left to right) is such that negative class is first and positive class is second.
`nfac`	Number of factors for each Parafac model to estimate. Default is nfac = 1.
`nfolds`	Numeric setting number of folds for k-fold cross-validation. Must be 2 or greater. Default is 10 folds.
`foldid`	Integer vector containing fold IDs for k-fold cross-validation. If not provided, fold IDs are generated randomly for number of folds 'nfolds'.
`prior`	Prior probabilities of class membership. If unspecified, the class proportions for input 'y' are used. If present, the probabilities should be specified in the order of the factor levels of input 'y'.
`method`	Character vector indicating classification methods to use. Possible methods include penalized logistic regression (PLR), support vector machine (SVM), and random forest (RF). If none selected, default is to use all methods. See example.
`family`	Character value specifying binary classification (`family = "binomial"`) or multiclass classification (`family = "multinomial"`). If not provided, number of levels of input 'y' is used, where two levels is binary, and where three or more levels is multiclass.
`alpha`	Values for penalized logistic regression alpha parameter; default is alpha = seq(0, 1, length = 6). Must be numeric and contain only real numbers between 0 and 1, inclusive.

`lambda`	Optional user-supplied lambda sequence for `cv.glmnet`. Default is NULL.
`cost`	Values for support vector machine cost parameter; default is cost = c(1, 2, 4, 8, 16, 32, 64). Must be numeric and contain only real numbers greater than or equal to zero.
`gamma`	Values for support vector machine gamma parameter; default is gamma = c(0, 0.01, 0.1, 1, 10, 100, 1000). Must be numeric and greater than or equal to 0.
`ntree`	Values for random forest number of trees parameter; default is ntree = c(100, 200, 400, 600, 800, 1600, 3200). Must be numeric and contain only integers greater than or equal to 1.
`nodesize`	Values for random forest node size parameter; default is nodesize = c(1, 2, 4, 8, 16, 32, 64). Must be numeric and contain only integers greater than or equal to 1.
`size`	Values for single-layer neural network size parameter; default is size = c(1, 2, 4, 8, 16, 32, 64). Must be numeric and contain only integers greater than or equal to 0.
`decay`	Values for neural network decay parameter; default is decay = c(0.001, 0.01, 0.1, 1, 2, 4, 8, 16). Must be numeric and contain only real numbers.
`parallel`	Logical indicating if package `parallel` should be used for parallel computing. For support vector machine and random forest, `doParallel` package is used as a wrapper. Defaults to FALSE, which implements sequential computing.
`cl`	Cluster for parallel computing, which is used when `parallel = TRUE`. Note that if `parallel = TRUE` and `cl = NULL`, then the cluster is defined as makeCluster(detectCores()).
`verbose`	If TRUE, progress is printed.
`cmode`	Integer value of 1, 2, or 3 (or 4 if 'x' is a four-way array) specifying mode whose factor weights will be predictors for classification. Defaults to last mode of inputted array (i.e. 3 for three-way array, and 4 for four-way array).
`...`	Additional arguments to be passed to function 'parafac' for Parafac model estimation.

Details

After fitting a Parafac model with package `multiway` (see `parafac` in `multiway` for details), estimated classification mode weight matrix is passed to one or several of four classification methods– including penalized logistic regression (PLR); support vector machine (SVM); random forest (RF); and feed-forward neural network (NN).

Package `glmnet` fits models for PLR. PLR tunes penalty parameter lambda; elastic net parameter alpha is set by user (see `cv.glmnet` in package `glmnet`). For SVM, package `e1071` is used with a radial basis kernel. Penalty parameter cost and radial basis parameter gamma are used (see `svm` in package `e1071`). For RF, package `randomForest` is used and implements Breiman's random forest algorithm. Number of predictors sampled at each node split is set at default of sqrt(R), where R is the number of Parafac factors. Two tuning parameters allowed are ntree, the number of trees to be grown, and nodesize, the minimum size of terminal nodes (see `randomForest` in package `randomForest`). For NN, package `nnet` fits a single-hidden-layer, feed-forward neural network model. Penalty parameters size (number of hidden layer units) and decay (weight decay) are used (see `nnet`).

For all four methods, k-fold cross-validation is implemented to tune classification parameters where the number of folds is set by argument `nfolds`.

Value

Returns an object of class 'cpfa' with the following elements:

`opt.model`	List containing optimal model for tuned classification methods for each Parafac model estimated.
`opt.param`	Data frame containing optimal parameters for tuned classification methods.
`kcv.error`	Data frame containing KCV misclassification error for optimal parameters for tuned classification methods.
`est.time`	Data frame containing estimation times for fitting Parafac model and for tuning classification methods.
`method`	Numeric indicating classification methods used. Value of '1' indicates PLR; value of '2' indicates SVM; and value of '3' indicates 'RF'.
`x`	three-way or four-way array used.
`y`	Factor containing class labels used. Note that output `y` is recoded such that the input labels of `y` are converted to numeric integers from 0 through the number of levels, which are then applied as labels for output `y`.
`Aweights`	List containing estimated A weights for each Parafac model estimated.
`Bweights`	List containing estimated B weights for each Parafac model estimated.
`Cweights`	List containing estimated C weights for each Parafac model estimated. Null if inputted argument `x` was a 3 array.
`const`	Constraints used in Parafac model estimation. If argument `const` was not inputted, output value will be unconstrained for all modes.
`cmode`	Integer value of 1, 2, or 3 (or 4 if `x` is a four-way array) specifying mode whose factor weights were predictors for classification.
`family`	Character value specifying whether classification was binary (`family = "binomial"`) or multiclass (`family = "multinomial"`).

Note

If argument `cmode` is not null, input array `x` is reshaped with function `aperm` such that the `cmode` dimension of `x` is ordered last. Estimated mode A and B (and mode C for a four-way array `x`) weights that are outputted as `Aweights` and `Bweights` (and `Cweights`) reflect this permutation. For example, if `x` is a four-way array and `cmode = 2`, original modes/dimensions 1, 2, 3, and 4 will correspond to output order 1, 3, 4, 2. Here, output A = input 1; B = 3, and C = 4 (i.e. second mode specified by `cmode` has been moved to the last/D mode).

Author(s)

Matthew A. Snodgress <snodg031@umn.edu>

References

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297.

Friedman, J. Hastie, T., and Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22.

Harshman, R. A. (1970). Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. UCLA Working Papers in Phonetics, 16, 1-84.

Harshman, R. A. and Lundy, M. E. (1994). PARAFAC: Parallel factor analysis. Computational Statistics and Data Analysis, 18, 39-72.

Helwig, N. E. (2017). Estimating latent trends in multivariate longitudinal data via Parafac2 with functional and structural constraints. Biometrical Journal, 59(4), 783-803.

Helwig, N. E. (2019). multiway: Component Models for Multi-Way Data. R package version 1.0-6.

Liaw, A. and Wiener, M. (2002). Classification and Regression by randomForest. R News 2(3), 18–22.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2021). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.7-6.

Ripley, B. (1994). Neural networks and related methods for classification. Journal of the Royal Statistical Society: Series B (Methodological), 56(3), 409-437.

Venables, W. and Ripley, B. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2), 301-320.

Examples

```
########## Random binary example with 3-way array ##########

# create random data array with Parafac structure
set.seed(3)
mydim <- c(60, 16, 80)
nf <- 3
Amat <- matrix(rnorm(mydim[1]*nf), nrow = mydim[1], ncol = nf)
Bmat <- matrix(runif(mydim[2]*nf), nrow = mydim[2], ncol = nf)
Cmat <- matrix(runif(mydim[3]*nf), nrow = mydim[3], ncol = nf)
Xmat <- tcrossprod(Amat, krprod(Cmat, Bmat))
Xmat <- array(Xmat, dim = mydim)
Emat <- array(rnorm(prod(mydim)), dim = mydim)
X <- Xmat + Emat
y <- factor(rbinom(mydim[3], 1, 0.4))

# initialize
alpha <- seq(0, 1, length = 2)
gamma <- c(0, 0.01)
cost <- c(1, 2)
ntree <- c(100, 200)
```

```
nodesize <- c(1, 2)
size <- c(1, 2)
decay <- c(0, 1)
method <- c("PLR", "SVM", "RF", "NN")
nfolds <- 3
const <- c("orthog", "uncons", "uncons")
foldid <- sample(rep(1:nfolds, length.out = length(y)))

# estimate Parafac models and use third mode to tune classification methods
tune.object <- cpfa(x = X, y = y, nfac = 3, nfolds = nfolds,
                         foldid = foldid, method = method, alpha = alpha,
                         gamma = gamma, cost = cost, ntree = ntree,
                         nodesize = nodesize, size = size, decay = decay,
                         parallel = FALSE, const = const)
print.cpfa(tune.object)

# Note: see package vignette for more examples
```

---

cpm — *Classification Performance Measures*

Description

Calculates multiple performance measures for binary or multiclass classification. Uses known class labels and evaluates against predicted labels.

Usage

```
cpm(x, y, level = NULL, fbeta = NULL, prior = NULL)
```

Arguments

`x`	Known class labels of class numeric, factor, or integer. If factor, converted to class integer in order of factor levels with integers beginning at 0 (i.e. for binary classification, factor levels become 0 and 1; for multiclass, 0, 1, 2, etc.).
`y`	Predicted class labels of class numeric, factor, or integer. If factor, converted to class integer in order of factor levels with integers beginning at 0 (i.e. for binary classification, factor levels become 0 and 1; for multiclass, 0, 1, 2, etc.).
`level`	Optional argument specifying possible class labels. For cases when `x` or `y` do not contain all possible classes. Can be of class numeric, integer, or character. Must contain two elements for binary classification, and contain three or more elements for multiclass classification. If integer, integers should be ordered (e.g. `c(0, 1)`; or `c(0, 1, 2)`). Note: if both `x` and `y` jointly contain only a single value (e.g. 1), must specify argument `level` in order to identify classification as binary or multiclass.
`fbeta`	Optional numeric argument specifying beta value for F-score. Defaults to `fbeta = 1`, providing an F1-score (balanced harmonic mean between precision and recall).

`prior`	Optional numeric argument specifying weights for classes. Defaults to `prior = c(rep(1/llev, llev))`, where 'llev' is the number of classes, providing equal importance across classes.

Details

Selecting one class as a negative class and one class as positive, binary classification generates four possible outcomes: (1) negative cases classified as positives, called false positives (FP); (2) negative cases classified as negatives, called true negatives (TN); (3) positive cases classified as negatives, called false negatives (FN); and (4) positive cases classified as positives, called true positives (TP).

Multiple evaluation measures are calculated using these four outcomes. Measures include: overall error (ERR), also called fraction incorrect; overall accuracy (ACC), also called fraction correct; true positive rate (TPR), also called recall, hit rate, or sensitivity; false negative rate (FNR), also called miss rate; false positive rate (FPR), also called fall-out; true negative rate (TNR), also called specificity or selectivity; positive predictive value (PPV), also called precision; false discovery rate (FDR); negative predictive value (NPV); false omission rate (FOR); and F-score (FS).

In multiclass classification, the four outcomes are possible for each individual class in macro-averaging, and evaluation measures are averaged over classes. Macro-averaging gives equal importance to all classes. For multiclass classification, calculated measures are currently only macro-averaged.

Note that binary classification assumes a positive class and negative class (i.e. contains a reference group) and is ordered. Multiclass classification is currently unordered.

Computational details:

ERR = (FP + FN) / (TP + TN + FP + FN).

ACC = (TP + TN) / (TP + TN + FP + FN), and ACC = 1 - ERR.

TPR = TP / (TP + FN).

FNR = FN / (FN + TP), and FNR = 1 - TPR.

FPR = FP / (FP + TN).

TNR = TN / (TN + FP), and TNR = 1 - FPR.

PPV = TP / (TP + FP).

FDR = FP / (FP + TP), and FDR = 1 - PPV.

NPV = TN / (TN + FN).

FOR = FN / (FN + TN), and FOR = 1 - NPV.

FS = (1 + beta^2) * ((PPV * TPR) / (((beta^2)*PPV) + TPR)).

All evaluation measures calculated are between 0 and 1, inclusive. For multiclass classification, macro-averaged values are provided for each performance measure. Note that 'beta' in FS represents the relative weight such that recall (TPR) is beta times more important than precision (PPV). See reference in help document for more details.

Value

Returns list where first element is a full confusion matrix `cm` and second element is a data frame containing performance measures. Note that for multiclass classification, macro-averaged values are provided (each measure calculated for each class, then averaged over all classes; average is weighted by argument `prior` if provided). Excluding `cm`, second list element contains following performance measures:

`cm`	A confusion matrix with counts for each of possible outcomes.
`err`	Overall error (ERR). Also called fraction incorrect.
`acc`	Overall accuracy (ACC). Also called fraction correct.
`tpr`	True positive rate (TPR). Also called recall, hit rate, or sensitivity.
`fpr`	False positive rate (FPR). Also called fall-out.
`tnr`	True negative rate (TNR). Also called specificity or selectivity.
`fnr`	False negative rate (FNR). Also called miss rate.
`ppv`	Positive predictive value (PPV). Also called precision.
`npv`	Negative predicted value (NPV).
`fdr`	False discovery rate (FDR).
`fom`	False omission rate (FOR).
`fs`	F-score. Mean between TPR (recall) and PPV (precision) varying by importance given to recall over precision (see Details section and argument `fbeta`).

Author(s)

Matthew Snodgress <snodg031@umn.edu>

References

Sokolova, M., and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing and Management, 45(4), 427-437.

Examples

```
########## Random binary example with 3-way array ##########

# create random data array with Parafac structure
set.seed(3)
mydim <- c(60, 16, 80)
nf <- 3
Amat <- matrix(rnorm(mydim[1]*nf), nrow = mydim[1], ncol = nf)
Bmat <- matrix(runif(mydim[2]*nf), nrow = mydim[2], ncol = nf)
Cmat <- matrix(runif(mydim[3]*nf), nrow = mydim[3], ncol = nf)
Xmat <- tcrossprod(Amat, krprod(Cmat, Bmat))
Xmat <- array(Xmat, dim = mydim)
Emat <- array(rnorm(prod(mydim)), dim = mydim)
X <- Xmat + Emat
y <- factor(rbinom(mydim[3], 1, 0.4))

# initialize
alpha <- seq(0, 1, length = 2)
gamma <- c(0, 0.01)
cost <- c(1, 2)
ntree <- c(100, 200)
```

```
nodesize <- c(1, 2)
size <- c(1, 2)
decay <- c(0, 1)
method <- c("PLR", "SVM", "RF", "NN")
nfolds <- 3
const <- c("orthog", "uncons", "uncons")
foldid <- sample(rep(1:nfolds, length.out = length(y)))

# estimate Parafac models and use third mode to tune classification methods
tune.object <- cpfa(x = X, y = y, nfac = 3, nfolds = nfolds,
                         foldid = foldid, method = method, alpha = alpha,
                         gamma = gamma, cost = cost, ntree = ntree,
                         nodesize = nodesize, size = size, decay = decay,
                         parallel = FALSE, const = const)

# create random data array with Parafac structure and same A and B weights
mydim.new <- c(60, 16, 20)
Cmat <- matrix(runif(mydim.new[3]*nf), nrow = mydim.new[3], ncol = nf)
Xmat <- tcrossprod(Amat, krprod(Cmat, Bmat))
Xmat <- array(Xmat, dim = mydim.new)
Emat <- array(rnorm(prod(mydim.new)), dim = mydim.new)
Xnew <- Xmat + Emat

# predict class labels
predict.labels <- predict(object = tune.object, newdata = Xnew,
                                    type = "response")

# create new random class labels for two levels
newlabel <- as.numeric(factor(rbinom(mydim.new[3], 1, 0.4))) - 1

# calculate evaluation measure for PLR predicted
y.pred <- predict.labels[, 1]
evalmeasure <- cpm(x = newlabel, y = y.pred)
evalmeasure

# Note: see package vignette for more examples
```

`predict.cpfa` *Predict Method for Classification with Parallel Factor Analysis*

Description

`predict` method for class "cpfa".

Usage

```
## S3 method for class 'cpfa'
predict(object, newdata = NULL, nfac = NULL, method = NULL,
        type = c("response", "prob", "classify.weights"),
        threshold = NULL, ...)
```

Arguments

`object`	A fit object of class 'cpfa' from function 'cpfa'.
`newdata`	An optional three-way or four-way data array used to predict Parafac factor weights using estimated Parafac model factor weights from inputted object. Dimensions must match dimensions of original data for all modes except classification mode. If omitted, the original data are used.
`nfac`	Number of factors in models for prediction. Defaults to number of factors for each Parafac model from input 'object'.
`method`	Classification methods used for prediction. Defaults to methods used in input 'object'.
`type`	Character vector indicating type of prediction to return. Possible values include: (1) `"response"`, returning predicted class labels; (2) `"prob"`, returning predicted class probabilities; or (3) `"classify.weights"`, returning predicted factor weights used in classification from Parafac models specified. Defaults to `"response"`.
`threshold`	For binary classification, value indicating prediction threshold over which observations are classified as the positive class. If not provided, calculates threshold using class proportions in original data. For multiclass classification, `threshold` is not currently implemented.
`...`	additional arguments affecting the prediction produced (currently ignored).

Details

Predicts class labels for a binary or a multiclass outcome. Specifically, predicts factor weights for one mode of a Parallel Factor Analysis (Parafac) model-1 using newdata and previously estimated mode weights from original data. Passes predicted factor weights to one or several classification methods as new data for predicting class labels.

Tuning parameters optimized by k-fold cross-validation are used for each classification method (see help for `cpfa`). If not supplied in argument 'threshold', prediction threshold for all classification methods is calculated using proportions of class labels for original data in the binary case (and positive proportion is set as threshold). For multiclass case, class with highest probability is chosen. Calculates and returns only predicted probabilities without class assignment by specifying type = "prob". Returns only predicted factor weights if type = "classify.weights".

Value

Returns one of the following: (1) a data frame 'storfac'; (2) a list 'storprob', or (3) a list 'classify.weights':

`storfac`	A data frame containing predicted class labels or probabilities (binary case) for each Parafac model and classification method selected (see argument 'type'). Number of columns is equal to number of methods times number of Parafac models. Number of rows is equal to number of predicted observations.
`storprob`	A list containing predicted probabilities for each Parafac model and classification method selected (see argument 'type'). Only returned if original response was multiclass (3 or more class labels). Number of elements is equal to number of methods times number of Parafac models.

`cweights`	List containing predicted factor weights for each Parafac model. Length is equal to number of Parafac models estimated.

Author(s)

Matthew Snodgress <snodg031@umn.edu>

References

Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

Cortes, C., and Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

Friedman, J. Hastie, T., and Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22.

Harshman, R. A. (1970). Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. UCLA Working Papers in Phonetics, 16, 1-84.

Harshman, R. A. and Lundy, M. E. (1994). PARAFAC: Parallel factor analysis. Computational Statistics and Data Analysis, 18, 39-72.

Helwig, N. E. (2017). Estimating latent trends in multivariate longitudinal data via Parafac2 with functional and structural constraints. Biometrical Journal, 59(4), 783-803.

Helwig, N. E. (2019). multiway: Component Models for Multi-Way Data. R Package version 1.0-6.

Liaw, A., and Wiener, M. (2002). Classification and Regression by randomForest. R News 2(3), 18-22.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2021). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.7-6.

Ripley, B. (1994). Neural networks and related methods for classification. Journal of the Royal Statistical Society: Series B (Methodological), 56(3), 409-437.

Venables, W. and Ripley, B. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 67(2), 301-320.

Examples

```
########## Random binary example with 3-way array ##########

# create random data array with Parafac structure
set.seed(3)
mydim <- c(60, 16, 80)
nf <- 3
Amat <- matrix(rnorm(mydim[1]*nf), nrow = mydim[1], ncol = nf)
Bmat <- matrix(runif(mydim[2]*nf), nrow = mydim[2], ncol = nf)
Cmat <- matrix(runif(mydim[3]*nf), nrow = mydim[3], ncol = nf)
Xmat <- tcrossprod(Amat, krprod(Cmat, Bmat))
Xmat <- array(Xmat, dim = mydim)
Emat <- array(rnorm(prod(mydim)), dim = mydim)
X <- Xmat + Emat
```

```
y <- factor(rbinom(mydim[3], 1, 0.4))

# initialize
alpha <- seq(0, 1, length = 2)
gamma <- c(0, 0.01)
cost <- c(1, 2)
ntree <- c(100, 200)
nodesize <- c(1, 2)
size <- c(1, 2)
decay <- c(0, 1)
method <- c("PLR", "SVM", "RF", "NN")
nfolds <- 3
const <- c("orthog", "uncons", "uncons")
foldid <- sample(rep(1:nfolds, length.out = length(y)))

# estimate Parafac models and use third mode to tune classification methods
tune.object <- cpfa(x = X, y = y, nfac = 3, nfolds = nfolds,
                         foldid = foldid, method = method, alpha = alpha,
                         gamma = gamma, cost = cost, ntree = ntree,
                         nodesize = nodesize, size = size, decay = decay,
                         parallel = FALSE, const = const)

# create random data array with Parafac structure and same A and B weights
mydim.new <- c(60, 16, 20)
Cmat <- matrix(runif(mydim.new[3]*nf), nrow = mydim.new[3], ncol = nf)
Xmat <- tcrossprod(Amat, krprod(Cmat, Bmat))
Xmat <- array(Xmat, dim = mydim.new)
Emat <- array(rnorm(prod(mydim.new)), dim = mydim.new)
Xnew <- Xmat + Emat

# predict class labels
predict.labels <- predict(object = tune.object, newdata = Xnew,
                                    type = "response")
head(predict.labels)

# Note: see package vignette for more examples
```

Index