Package ‘classmap’

July 25, 2021

Type Package
Title Visualizing Classification Results
Date 2021-07-24
Version 1.1.1
Author Jakob Raymaekers [aut, cre],
       Peter Rousseeuw [aut]
Depends R (>= 3.5.0)
Suggests knitr, reshape2, svd, rpart.plot, nnet
Imports stats, graphics, ggplot2, robustbase, e1071, cellWise,
       cluster, kernlab, gridExtra, rpart, randomForest
Maintainer Jakob Raymaekers <jakob.raymaekers@kuleuven.be>
Description Tools to visualize the results of a classification of cases.
       The graphical displays include stacked plots, silhouette plots, quasi residual plots, and class maps.
       Implements the techniques described and illustrated in Raymaekers, Rousseeuw and Hubert (2021),
       Class maps for visualizing classification results, Technometrics, appeared online.
       <doi:10.1080/00401706.2021.1927849> (open access) and Raymaekers and Rousseeuw (2021),
       Silhouettes and quasi residual plots for neural nets and tree-based classifiers,
       <arXiv:2106.08814>. Examples can be found in the vignettes: \texttt{``Discriminant_analysis_examples'',``K_nearest_neighbors_examples'',}
       \texttt{``Support_vector_machine_examples'',``Rpart_examples'',``Random_forest_examples'',}
       and \texttt{``Neural_net_examples''}.
URL https://doi.org/10.1080/00401706.2021.1927849,
       https://arxiv.org/abs/2106.08814
License GPL (>= 2)
Encoding UTF-8
LazyData true
VignetteBuilder knitr
RoxygenNote 7.1.1
NeedsCompilation no
Repository CRAN
Date/Publication 2021-07-25 13:40:05 UTC
Draw the class map to visualize classification results.

Description

Draw the class map to visualize classification results, based on the output of one of the \texttt{vcr.***} functions in this package. The vertical axis of the class map shows each case’s PAC, the conditional probability that it belongs to an alternative class. The farness on the horizontal axis is the probability of a member of the given class being at most as far from the class as the case itself.

Usage

\begin{verbatim}
classmap(vcrout, whichclass, classLabels = NULL, classCols = NULL,
         main = NULL, cutoff = 0.99, plotcutoff = TRUE, identify = FALSE,
         cex = 1, cex.main=1.2, cex.lab=NULL, cex.axis=NULL, opacity = 1,
         squareplot = TRUE, maxprob = NULL, maxfactor=NULL)
\end{verbatim}
classmap

Arguments

vcrout output of vcr.*.train or vcr.*.newdata. Required.
whichclass the number or level of the class to be displayed. Required.
classLabels the labels (levels) of the classes. If NULL, they are taken from vcrout.
classCols a list of colors for the class labels. There should be at least as many as there are levels. If NULL the classCols are taken as 2, 3, 4, ...
main title for the plot.
cutoff cases with overall farness vcrout$ofarness > cutoff are flagged as outliers.
plotcutoff If true, plots the cutoff on the farness values as a vertical line.
identify if TRUE, invoke graphics::identify after plotting: left-click on a point to get its number, then ESC to exit.
cex passed on to graphics::plot.
cex.main same, for title.
cex.lab same, for labels on horizontal and vertical axes.
cex.axis same, for axes.
opacity determines opacity of plotted dots. Value between 0 and 1, where 0 is transparent and 1 is opaque.
squareplot If TRUE, makes the axes of the plot equally long.
maxprob draws the farness axis at least upto probability maxprob. If NULL, the limits are obtained automatically.
maxfactor if not NULL, a number slightly higher than 1 to increase the space at the right hand side of the plot, to make room for marking points.

Value

Executing the function plots the class map and returns

coordinates a matrix with 2 columns containing the coordinates of the plotted points. The first coordinate is the quantile of the farness probability. This makes it easier to add text next to interesting points. If identify = T, the attribute ids of coordinates contains the row numbers of the identified points in the matrix coordinates.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J.(2021). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. (link to open access pdf)
See Also

vcr.da.train, vcr.da.newdata,
vcr.knn.train, vcr.knn.newdata,
vcr.svm.train, vcr.svm.newdata,
vcr.rpart.train, vcr.rpart.newdata,
vcr.forest.train, vcr.forest.newdata,
vcr.neural.train, vcr.neural.newdata

Examples

vcrout = vcr.da.train(iris[,1:4], iris[,5])
classmap(vcrout, "setosa", classCols = 2:4) # tight class
classmap(vcrout, "versicolor", classCols = 2:4) # less tight
# The cases misclassified as virginica are shown in blue.
classmap(vcrout, "virginica", classCols = 2:4)
# The case misclassified as versicolor is shown in green.

# For more examples, we refer to the vignettes:
vignette("Discriminant_analysis_examples")
vignette("K_nearest_neighbors_examples")
vignette("Support_vector_machine_examples")
vignette("Rpart_examples")
vignette("Random_forest_examples")
vignette("Neural_net_examples")

confmat.vcr

Build a confusion matrix from the output of a function vcr.*.*.

Description

Build a confusion matrix from the output of a function vcr.*.*. Optionally, a separate column for outliers can be added to the confusion matrix.

Usage

confmat.vcr(vcrout, cutoff = 0.99, showClassNumbers = FALSE,
showOutliers = TRUE, silent = FALSE)

Arguments

vcrout output of vcr.*.train or vcr.*.newdata.
cutoff cases with overall farness vcrout$ofarness > cutoff are flagged as outliers.
showClassNumbers if TRUE, the row and column names are the number of each level instead of the level itself. Useful for long level names.
showOutliers if TRUE and some points were flagged as outliers, it adds an extra column on the right of the confusion matrix for these outliers, with label "outl".
silent if FALSE, the confusion matrix and accuracy are shown on the screen.
Value

A confusion matrix

Author(s)

Raymaekers J., Rousseeuw P.J.

References


See Also

vcr.da.train, vcr.da.newdata,
vcr.knn.train, vcr.knn.newdata,
vcr.svm.train, vcr.svm.newdata,
vcr.rpart.train, vcr.rpart.newdata,
vcr.forest.train, vcr.forest.newdata,
vcr.neural.train, vcr.neural.newdata

Examples

vcrout = vcr.knn.train(scale(iris[,1:4]), iris[,5], k=5)
# The usual confusion matrix:
confmat.vcr(vcrout, showOutliers = FALSE)

# Cases with ofarness > cutoff are flagged as outliers:
confmat.vcr(vcrout, cutoff = 0.98)

# With the default cutoff = 0.99 only one case is flagged here:
confmat.vcr(vcrout)
# Note that the accuracy is computed before any cases
# are flagged, so it does not depend on the cutoff.

confmat.vcr(vcrout,showClassNumbers=TRUE)
# Shows class numbers instead of labels. This option can
# be useful for long level names.

# For more examples, we refer to the vignettes:
vignette("Discriminant_analysis_examples")
vignette("K_nearest_neighbors_examples")
vignette("Support_vector_machine_examples")
Amazon book reviews data

Description

This is a subset of the data used in the paper, which was assembled by Prettenhofer and Stein (2010). It contains 1000 reviews of books on Amazon, of which 500 were selected from the original training data and 500 from the test data.

The full dataset has been used for a variety of things, including classification using svm. The subset was chosen small enough to keep the computation time low, while still containing the examples in the paper.

Usage

data("data_bookReviews")

Format

A data frame with 1000 observations on the following 2 variables.

- **review**: the review in text format (character)
- **sentiment**: factor indicating the sentiment of the review: negative (1) or positive (2)

Source


Examples

data(data_bookReviews)

# Example review:
data_bookReviews[5, 1]

# The data are used in:
vignette("Support_vector_machine_examples")
data_floralbuds  Floral buds data

Description
This data on floral pear bud detection was first described by Wouters et al. The goal is to classify the instances into buds, branches, scales and support. The numeric vectors resulted from a multispectral vision sensor and describe the scanned images.

Usage
data("data_floralbuds")

Format
A data frame with 550 observations on the following 7 variables.

X1 numeric vector
X2 numeric vector
X3 numeric vector
X4 numeric vector
X5 numeric vector
X6 numeric vector
y a factor with levels branch bud scales support

Source

Examples
data("data_floralbuds")
str(data_floralbuds)
summary(data_floralbuds)

# The data are used in:
vignette("Discriminant_analysis_examples")
data_instagram

Description

This dataset contains information on fake (spam) accounts on Instagram. The original source is https://www.kaggle.com/free4ever1/instagram-fake-spammer-genuine-accounts by Bardiya Bakhshan-deh.

The data contains information on 696 Instagram accounts. For each account, 11 variables were recorded describing its characteristics. The goal is to detect fake instagram accounts, which are used for spamming.

Usage

data("data_instagram")

Format

A data frame with 696 observations on the following variables.

- **profile.pic**  binary, indicates whether profile has picture.
- **nums.length.username**  ratio of number of numerical chars in username to its length.
- **fullname.words**  number of words in full name.
- **nums.length.fullname**  ratio of number of numerical characters in full name to its length.
- **name.username**  binary, indicates whether the name and username of the profile are the same.
- **description.length**  length of the description/biography of the profile (in number of characters).
- **external.URL**  binary, indicates whether profile has external url.
- **private**  binary, indicates whether profile is private or not.
- **X.posts**  number of posts made by profile.
- **X.followers**  number of followers.
- **X.follows**  numbers of follows.
- **y**  whether profile is fake or not.
- **dataType**  vector taking the values “train” or “test” indicating whether the observation belongs to the training or the test data.

Source

https://www.kaggle.com/free4ever1/instagram-fake-spammer-genuine-accounts

Examples

data(data_instagram)
str(data_instagram)

# The data are used in:
vignette("Random_forest_examples")
Description

This dataset contains information on 1309 passengers of the RMS Titanic. The goal is to predict survival based on 11 characteristics such as the travel class, age and sex of the passengers.

The original data source is https://www.kaggle.com/c/titanic/data

The data is split up in a training data consisting of 891 observations and a test data of 418 observations. The response in the test set was obtained by combining information from other data files, and has been verified by submitting it as a ‘prediction’ to kaggle and getting perfect marks.

Usage

data("data_titanic")

Format

A data frame with 1309 observations on the following variables.

- **PassengerId**  a unique identified for each passenger.
- **Pclass**  travel class of the passenger.
- **Name**  name of the passenger.
- **Sex**  sex of the passenger.
- **Age**  age of the passenger.
- **SibSp**  number of siblings and spouses traveling with the passenger.
- **Parch**  number of parents and children traveling with the passenger.
- **Ticket**  Ticket number of the passenger.
- **Fare**  fare paid for the ticket.
- **Cabin**  cabin number of the passenger.
- **Embarked**  Port of embarkation. Takes the values C (Cherbourg), Q (Queenstown) and S (Southampton).
- **y**  factor indicating casualty or survivor.
- **dataType**  vector taking the values “train” or “test” indicating whether the observation belongs to the training or the test data.

Source

https://www.kaggle.com/c/titanic/data
Examples

data("data_titanic")
traindata = data_titanic[which(data_titanic$DataType == "train"), -13]
testdata = data_titanic[which(data_titanic$DataType == "test"), -13]
str(traindata)
table(traindata$y)

# The data are used in:
vignette("Rpart_examples")

makeFV

Constructs feature vectors from a kernel matrix.

Description

Constructs feature vectors from a kernel matrix.

Usage

makeFV(kmat, transfmat=NULL, precS=1e-12)

Arguments

kmat

a kernel matrix. If transfmat is NULL, we are dealing with training data and then kmat must be a square kernel matrix (of size \( n \) by \( n \) when there are \( n \) cases). Such a PSD matrix kmat can e.g. be produced by makeKernel or by kernlab:::kernelMatrix. If on the other hand transfmat is not NULL, we are dealing with a test set. See details for the precise working.

transfmat

transformation matrix. If not NULL, it is the value transfmat of makeFV on training data. It has to be a square matrix, with as many rows as there were training data.

precS

if not NULL, eigenvalues of kmat below precS will be set equal to precS.

Details

If transfmat is non-NULL, we are dealing with a test set. Denote the number of cases in the test set by \( m \geq 1 \). Each row of kmat of the test set then must contain the kernel values of a new case with all cases in the training set. Therefore the kernel matrix kmat must have dimensions \( m \) by \( n \). The matrix kmat can e.g. be produced by makeKernel. It can also be obtained by running kernlab:::kernelMatrix on the union of the training set and the test set, yielding an \( (n + m) \) by \( (n + m) \) matrix, from which one then takes the \([ (n + 1) : m, 1 : n ]\) submatrix.
makeFV

Value

A list with components:

- **Xf**: When makeKV is applied to the training set, Xf has coordinates of n points (vectors), the plain inner products of which equal the kernel matrix of the training set. That is, kmat = Xf Xf'. The Xf are expressed in an orthogonal basis in which the variance of the coordinates is decreasing, which is useful when plotting the first few coordinates. When makeFV is applied to a test set, Xf are coordinates of the feature vectors of the test set in the same space as those of the training set, and then kmat = Xf Xf_training'.

- **transfmat**: square matrix for transforming kmat to Xf. The actual transformation needs to be carried out by makeFV because it is not a simple matrix product.

Author(s)

Raymaekers J., Rousseeuw P.J.

References


See Also

makeKernel

Examples

```r
library(e1071)
set.seed(1); X = matrix(rnorm(200*2),ncol=2)
X[1:100,] = X[1:100,]+2
X[101:150,] = X[101:150,-2]
y = as.factor(c(rep("blue",150),rep("red",50)))
cols = c("deepskyblue3","red")
plot(X,col=cols[as.numeric(y)],pch=19)
# We now fit an SVM with radial basis kernel to the data:
svmfit = svm(y~., data=data.frame(X=X,y=y), scale=FALSE,
kernel="radial", cost=10, gamma=1, probability=TRUE)
Kxx = makeKernel(X,svfit=svmfit)
outFV = makeFV(Kxx)
Xf = outFV$Xf # The data matrix in this feature space.
dim(Xf) # The feature vectors are high dimensional.
# The inner products of Xf match the kernel matrix:
max(abs(as.vector(Kxx - crossprod(t(Xf), t(Xf))))) # 6.167711e-11 # tiny, OK
range(rowSums(Xf^2)) # all points in Xf lie on the unit sphere.
pairs(Xf[,1:5],col=cols[as.numeric(y)])
# In some of these we see spherical effects, e.g.
plot(Xf[,11],Xf[,5],col=cols[as.numeric(y)],pch=19)
# The data look more separable here than in the original # two-dimensional space.
```
# For more examples, we refer to the vignette:

vignette("Support_vector_machine_examples")

---

## makeKernel

### Compute kernel matrix

#### Description

Computes kernel value or kernel matrix, where the kernel type is extracted from an svm trained by `e1071::svm`.

#### Usage

```r
makeKernel(X1, X2 = NULL, svfit)
```

#### Arguments

- `X1`: first matrix (or vector) of coordinates.
- `X2`: if not `NULL`, second data matrix or vector. If `NULL`, `X2` is assumed equal to `X1`.
- `svfit`: output from `e1071::svm`.

#### Value

the kernel matrix, of dimensions `nrow(X1)` by `nrow(X2)`. When both `X1` and `X2` are vectors, the result is a single number.

#### Author(s)

Raymaekers J., Rousseeuw P.J.

#### References


#### See Also

`makeFV`
Examples

```r
library(e1071)
set.seed(1); X = matrix(rnorm(200*2),ncol=2)
X[1:100,] = X[1:100,]+2
X[101:150,] = X[101:150,]-2
y = as.factor(c(rep("blue",150),rep("red",50))) # two classes
# We now fit an SVM with radial basis kernel to the data:
set.seed(1) # to make the result of svm() reproducible.
svmfit = svm(y~., data=data.frame(X=X,y=y), scale=FALSE,
              kernel="radial", cost=10, gamma=1, probability=TRUE)
Kxx = makeKernel(X,svfit=svmfit)
# The result is a square kernel matrix:
dim(Kxx) # 200 200
Kxx[1:5,1:5]

# For more examples, we refer to the vignette:
vignette("Support_vector_machine_examples")
```

---

**silplot**  
*Draw the silhouette plot of a classification*

**Description**

Draw the silhouette plot to visualize classification results, based on the output of one of the vcr.*.* functions in this package. The horizontal axis of the silhouette plot shows each case’s s(i).

**Usage**

```r
silplot(vcrout, classLabels = NULL, classCols = NULL,
        showLegend = TRUE, showClassNumbers = FALSE,
        showCases = FALSE, drawLineAtAverage = FALSE,
        topdown = TRUE, main = NULL, summary = TRUE)
```

**Arguments**

- `vcrout` output of vcr.*.train or vcr.*.newdata. Required.
- `classLabels` the labels (levels) of the classes. If NULL, they are taken from vcrout.
- `classCols` a list of colors for the classes. There should be at least as many as there are levels. If NULL a default palette is used.
- `showLegend` if TRUE, a legend is shown to the right of the plot.
- `showClassNumbers` if TRUE, the legend will show the class numbers instead of the class labels.
- `showCases` if TRUE, the plot shows the numbers of the cases. They are only readable when the number of cases is relatively small.
- `topdown` if TRUE (the default), the silhouettes are plotted from top to bottom. Otherwise they are plotted from left to right.
stackedplot

Make a vertically stacked mosaic plot of class predictions.

Description

Make a vertically stacked mosaic plot of class predictions from the output of `vcr.*.train` or `vcr.*.newdata`. Optionally, the outliers for each class can be shown as a gray rectangle at the top.
Usage

```r
stackedplot(vcrout, cutoff = 0.99, classCols = NULL, classLabels = NULL,
            separSize=1, minSize=1.5, showOutliers = TRUE, showLegend = FALSE,
            htitle = NULL, vtitle = NULL)
```

Arguments

- **vcrout**: output of `vcr.*.train` or `vcr.*.newdata`.
- **cutoff**: cases with overall farness `vcrout$ofarness > cutoff` are flagged as outliers.
- **classCols**: user-specified colors for the classes. If `NULL` a default palette is used.
- **classLabels**: names of given labels. If `NULL` they are taken from `vcrout`.
- **separSize**: how much white between rectangles.
- **minSize**: rectangles describing less than `minSize` percent of the data, are shown as `minSize` percent.
- **showOutliers**: if `TRUE`, shows a separate class in gray with the outliers, always at the top.
- **showLegend**: if `TRUE`, a legend is shown to the right of the plot. Default `FALSE`, since the legend is not necessary as the colors are already visible in the bottom part of each stack.
- **htitle**: title for horizontal axis (given labels). If `NULL`, a default title is shown.
- **vtitle**: title for vertical axis (predicted labels). If `NULL`, a default title is shown.

Value

A ggplot object.

Author(s)

Raymaekers J., Rousseeuw P.J.

References


See Also

- `vcr.da.train`, `vcr.da.newdata`
- `vcr.knn.train`, `vcr.knn.newdata`
- `vcr.svm.train`, `vcr.svm.newdata`
- `vcr.rpart.train`, `vcr.rpart.newdata`
- `vcr.forest.train`, `vcr.forest.newdata`
- `vcr.neural.train`, `vcr.neural.newdata`
Examples

data("data_floralbuds")
X = data_floralbuds[,1:6]; y = data_floralbuds[,7]
vcrout = vcr.da.train(X,y)
cols = c("saddlebrown", "orange", "olivedrab4", "royalblue3")
stackedplot(vcrout, classCols = cols, showLegend=TRUE)

# The legend is not really needed, since we can read the
# color of a class from the bottom of its vertical bar:
stackedplot(vcrout, classCols = cols)

# If we do not wish to show outliers:
stackedplot(vcrout, classCols = cols, showOutliers=FALSE)

# For more examples, we refer to the vignettes:
vignette("Discriminant_analysis_examples")
vignette("K_nearest_neighbors_examples")
vignette("Support_vector_machine_examples")
vignette("Rpart_examples")
vignette("Random_forest_examples")
vignette("Neural_net_examples")

---

vcr.da.newdata  

*Carry out discriminant analysis on new data, and prepare to visualize its results.*

Description

Predicts class labels for new data by discriminant analysis, using the output of `vcr.da.train` on the training data. For new data cases whose label in `yintnew` is non-missing, additional output is produced for constructing graphical displays such as the `classmap`.

Usage

```
vcr.da.newdata(Xnew, ynew=NULL, vcr.da.train.out)
```

Arguments

- **Xnew**
  - data matrix of the new data, with the same number of columns as in the training data. Missing values are not allowed.

- **ynew**
  - factor with class membership of each new case. Can be `NA` for some or all cases. If `NULL`, is assumed to be `NA` everywhere.

- **vcr.da.train.out**
  - output of `vcr.da.train` on the training data.
vcr.da.newdata

Value

A list with components:

- **yintnew**: number of the given class of each case. Can contain NA's.
- **ynew**: given class label of each case. Can contain NA’s.
- **levels**: levels of the response, from vcr.da.train.out.
- **predint**: predicted class number of each case. Always exists.
- **pred**: predicted label of each case.
- **altint**: number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose ynew is missing.
- **altlab**: label of the alternative class. Is NA for cases whose ynew is missing.
- **PAC**: probability of the alternative class. Is NA for cases whose ynew is missing.
- **fig**: distance of each case $i$ to each class $g$. Always exists.
- **farness**: farness of each case $i$ from its given class. Is NA for cases whose ynew is missing.
- **ofarness**: For each case $i$, its lowest fig[i,g] to any class $g$. Always exists.
- **classMS**: list with center and covariance matrix of each class, from vcr.da.train.out.
- **lCurrent**: log of mixture density of each case in its given class. Is NA for cases with missing ynew.
- **lPred**: log of mixture density of each case in its predicted class. Always exists.
- **lAlt**: log of mixture density of each case in its alternative class. Is NA for cases with missing ynew.

Author(s)

Raymaekers J., Rousseeuw P.J.

References


See Also

vcr.da.train, classmap, silplot, stackedplot

Examples

vcr.train = vcr.da.train(iris[,1:4], iris[,5])
inds = c(51:150) # a subset, containing only 2 classes
iris2 = iris[inds,] # fake "new" data
iris2[c(1:10,51:60),5] = NA
vcr.test = vcr.da.newdata(iris2[,1:4],iris2[,5],vcr.train)
vcr.test$PAC[1:25] # between 0 and 1. Is NA where the response is.
plot(vcr.test$PAC, vcr.train$PAC[inds]); abline(0,1) # match
plot(vcr.test$farness, vcr.train$farness[inds]); abline(0,1) # match
confmat.vcr(vcr.train) # for comparison
confmat.vcr(vcr.test)
stackedplot(vcr.train) # for comparison
stackedplot(vcr.test)
classmap(vcr.train, "versicolor", classCols = 2:4) # for comparison
classmap(vcr.test, "versicolor", classCols = 2:4) # has fewer points

# For more examples, we refer to the vignette:
vignette("Discriminant_analysis_examples")

vcr.da.train

 Carry out discriminant analysis on training data, and prepare to visualize its results.

Description

Custom DA function which prepares for graphical displays such as the classmap. The discriminant analysis itself is carried out by the maximum a posteriori rule, which maximizes the density of the mixture.

Usage

vcr.da.train(X, y, rule = "QDA", estmethod = meancov)

Arguments

x  a numerical matrix containing the predictors in its columns. Missing values are not allowed.
y  a factor with the given class labels.
rule  either "QDA" for quadratic discriminant analysis or "LDA" for linear discriminant analysis.
estmethod  function for location and covariance estimation. Should return a list with the center $m$ and the covariance matrix $S$. The default is meancov (classical mean and covariance matrix), and the option DetMCD (based on robustbase::covMcd) is also provided.

Value

A list with components:

yint  number of the given class of each case. Can contain NA’s.
y  given class label of each case. Can contain NA’s.
levels  levels of y
predint  predicted class number of each case. For each case this is the class with the highest posterior probability. Always exists.
pred  predicted label of each case.
altint  number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose y is missing.
altlab  label of the alternative class. Is NA for cases whose y is missing.
PAC  probability of the alternative class. Is NA for cases whose y is missing.
figparams  parameters for computing fig. can be used for new data.
fig  distance of each case i from each class g. Always exists.
farness  farness of each case i from its given class. Is NA for cases whose y is missing.
ofarness  for each case i, its lowest fig[i,g] to any class g. Always exists.
classMS  list with center and covariance matrix of each class
lCurrent  log of mixture density of each case in its given class. Is NA for cases with missing y.
lPred  log of mixture density of each case in its predicted class. Always exists.
lAlt  log of mixture density of each case in its alternative class. Is NA for cases with missing y.

Author(s)
Raymaekers J., Rousseeuw P.J.

References

See Also
vcr.da.newdata, classmap, silplot, stackedplot

Examples

data("data_floralbuds")
X = data_floralbuds[,1:6]; y = data_floralbuds[,7]
vcrout = vcr.da.train(X,y,rule="QDA")
# For linear discriminant analysis, put rule="LDA".
confmat.vcr(vcrout) # There are a few outliers
cols = c("saddlebrown","orange","olivedrab4","royalblue3")
stackedplot(vcrout, classCols = cols)
classmap(vcrout, "bud", classCols = cols)

# For more examples, we refer to the vignettes:
vignette("Discriminant_analysis_examples")
vcr.forest.newdata

Prepare for visualization of a random forest classification on new data.

Description

Produces output for the purpose of constructing graphical displays such as the classmap on new data. Requires the output of vcr.forest.train as an argument.

Usage

vcr.forest.newdata(Xnew, ynew = NULL, vcr.forest.train.out, LOO = FALSE)

Arguments

Xnew  data matrix of the new data, with the same number of columns d as in the training data. Missing values are not allowed.

ynew  factor with class membership of each new case. Can be NA for some or all cases. If NULL, is assumed to be NA everywhere.

vcr.forest.train.out  output of vcr.forest.train on the training data.

LOO  leave one out. Only used when testing this function on a subset of the training data. Default is LOO=FALSE.

Value

A list with components:

- yintnew  number of the given class of each case. Can contain NA’s.
- ynew  given class label of each case. Can contain NA’s.
- levels  levels of the response, from vcr.forest.train.out.
- predint  predicted class number of each case. Always exists.
- pred  predicted label of each case.
- altint  number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose ynew is missing.
- altlab  alternative label if yintnew was given, else NA.
- PAC  probability of the alternative class. Is NA for cases whose ynew is missing.
- fig  distance of each case i from each class g. Always exists.
- farness  farness of each case from its given class. Is NA for cases whose ynew is missing.
- ofarness  for each case i, its lowest fig[i,g] to any class g. Always exists.
vcr.forest.train

Prepare for visualization of a random forest classification on training data

Description

Produces output for the purpose of constructing graphical displays such as the classmap and silplot. The user first needs to train a random forest on the data by randomForest::randomForest. This then serves as an argument to vcr.forest.train.

Usage

vcr.forest.train(X, y, trainfit, type = list(), k = 5, stand = TRUE)
Arguments

X  A rectangular matrix or data frame, where the columns (variables) may be of mixed type.

y  factor with the given class labels. It is crucial that X and y are exactly the same as in the call to randomForest::randomForest. y is allowed to contain NA’s.

trainfit  the output of a randomForest::randomForest training run.

k  the number of nearest neighbors used in the farness computation.

type  list for specifying some (or all) of the types of the variables (columns) in X, used for computing the dissimilarity matrix, as in cluster::daisy. The list may contain the following components: "ordratio" (ratio scaled variables to be treated as ordinal variables), "logratio" (ratio scaled variables that must be logarithmically transformed), "asymm" (asymmetric binary) and "symm" (symmetric binary variables). Each component’s value is a vector, containing the names or the numbers of the corresponding columns of X. Variables not mentioned in the type list are interpreted as usual (see argument X).

stand  whether or not to standardize numerical (interval scaled) variables by their range as in the original cluster::daisy code for the farness computation. Defaults to TRUE.

Value

A list with components:

X  The data used to train the forest.

yint  number of the given class of each case. Can contain NA’s.

y  given class label of each case. Can contain NA’s.

levels  levels of y

predint  predicted class number of each case. For each case this is the class with the highest posterior probability. Always exists.

pred  predicted label of each case.

altint  number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose y is missing.

altlab  label of the alternative class. Is NA for cases whose y is missing.

PAC  probability of the alternative class. Is NA for cases whose y is missing.

figparams  parameters for computing fig, can be used for new data.

fig  distance of each case i from each class g. Always exists.

farness  farness of each case from its given class. Is NA for cases whose y is missing.

ofarness  for each case i, its lowest fig[i,g] to any class g. Always exists.

trainfit  The trained random forest which was given as an input to this function.
vcr.knn.newdata

Author(s)
Raymaekers J., Rousseeuw P.J.

References
Raymaekers J., Rousseeuw P.J. (2021). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. (link to open access pdf)

See Also
vcr.forest.newdata, classmap, silplot, stackedplot

Examples
library(randomForest)
data("data_instagram")
traindata = data_instagram[which(data_instagram$DataType == "train"), -13]
set.seed(71) # randomForest is not deterministic
rfout = randomForest(y ~ ., data=traindata, keep.forest=TRUE)
mytype = list(syym = c(1, 5, 7, 8)) # These 4 columns are # (symmetric) binary variables. The variables that are not # listed are interval-scaled by default.
x_train = traindata[, -12]
y_train = traindata[, 12]
# Prepare for visualization:
vcrtrain = vcr.forest.train(X = x_train, y = y_train,
                           trainfit = rfout, type = mytype)
confmat.vcr(vcrtrain)
stackedplot(vcrtrain, classCols = c(4,2))
silplot(vcrtrain, classCols = c(4,2))
classmap(vcrtrain, "genuine", classCols = c(4,2))
classmap(vcrtrain, "fake", classCols = c(4,2))

---

vcr.knn.newdata

*Carry out a k-nearest neighbor classification on new data, and prepare to visualize its results.*

Description
Predicts class labels for new data by k nearest neighbors, using the output of vcr.knn.train on the training data. For cases in the new data whose given label ynew is not NA, additional output is produced for constructing graphical displays such as the classmap.

Usage
vcr.knn.newdata(Xnew = NULL, ynew=NULL, vcr.knn.train.out, LOO=FALSE)
vcr.knn.newdata

Arguments

Xnew
If the training data was a matrix of coordinates, Xnew must be such a matrix with the same number of columns. If the training data was a set of dissimilarities, Xnew must be a rectangular matrix of dissimilarities, with each row containing the dissimilarities of a new case to all training cases. Missing values are not allowed.

ynew
factor with class membership of each new case. Can be NA for some or all cases. If NULL, is assumed to be NA everywhere.

vcr.knn.train.out
output of vcr.knn.train on the training data.

LOO
leave one out. Only used when testing this function on a subset of the training data. Default is LOO=FALSE.

Value

A list with components:

yintnew
number of the given class of each case. Can contain NA’s.

ynew
given class label of each case. Can contain NA’s.

levels
levels of the response, from vcr.knn.train.out.

predint
predicted class number of each case. Always exists.

pred
predicted label of each case.

altint
number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose ynew is missing.

altlab
label of the alternative class. Is NA for cases whose ynew is missing.

PAC
probability of the alternative class. Is NA for cases whose ynew is missing.

fig
distance of each case i from each class g. Always exists.

farness
farness of each case from its given class. Is NA for cases whose ynew is missing.

ofarness
for each case i, its lowest fig[i, g] to any class g. Always exists.

k
the requested number of nearest neighbors, from vcr.knn.train.out.

ktrues
for each case this contains the actual number of elements in its neighborhood. This can be higher than k due to ties.

counts
a matrix with 3 columns, each row representing a case. For the neighborhood of each case it says how many members it has from the given class, the predicted class, and the alternative class. The first and third entry is NA for cases whose ynew is missing.

Author(s)

Raymaekers J., Rousseeuw P.J.
vcr.knn.train

Carry out a k-nearest neighbor classification on training data, and prepare to visualize its results.

Description

Carries out a k-nearest neighbor classification on the training data. Various additional output is produced for the purpose of constructing graphical displays such as the classmap.

Usage

vcr.knn.train(X, y, k)

Arguments

X
This can be a rectangular matrix or data frame of (already standardized) measurements, or a dist object obtained from stats::dist or cluster::daisy. Missing values are not allowed.

y
factor with the given (observed) class labels. There need to be non-missing y in order to be able to train the classifier.

k
the number of nearest neighbors used. It can be selected by running cross-validation using a different package.

Examples

data("data_floralbuds")
X = data_floralbuds[,1:6]; y = data_floralbuds[,7]
set.seed(12345); trainset = sample(1:550, 275)
vcr.train = vcr.knn.train(X[trainset,], y[trainset], k=5)
vcr.test = vcr.knn.newdata(X[-trainset,], y[-trainset], vcr.train)

confmat.vcr(vcr.train) # for comparison
confmat.vcr(vcr.test)

cols = c("saddlebrown", "orange", "olivedrab4", "royalblue3")
stackedplot(vcr.train, classCols = cols) # for comparison
stackedplot(vcr.test, classCols = cols)
classmap(vcr.train, "bud", classCols = cols) # for comparison
classmap(vcr.test, "bud", classCols = cols)

# For more examples, we refer to the vignette:
 vignette("K_nearest_neighbors_examples")

References


See Also

vcr.knn.train, classmap, silplot, stackedplot
**Value**

A list with components:

- `yint` number of the given class of each case. Can contain NA’s.
- `y` given class label of each case. Can contain NA’s.
- `levels` levels of `y`
- `predint` predicted class number of each case. Always exists.
- `pred` predicted label of each case.
- `altint` number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose `y` is missing.
- `altnlab` label of the alternative class. Is NA for cases whose `y` is missing.
- `PAC` probability of the alternative class. Is NA for cases whose `y` is missing.
- `figparams` parameters used to compute `fig`.
- `fig` distance of each case `i` from each class `g`. Always exists.
- `farness` farness of each case from its given class. Is NA for cases whose `y` is missing.
- `ofarness` for each case `i`, its lowest `fig[i,g]` to any class `g`. Always exists.
- `k` the requested number of nearest neighbors, from the arguments. Will also be used for classifying new data.
- `ktrues` for each case this contains the actual number of elements in its neighborhood. This can be higher than `k` due to ties.
- `counts` a matrix with 3 columns, each row representing a case. For the neighborhood of each case it says how many members it has from the given class, the predicted class, and the alternative class. The first and third entry is NA for cases whose `y` is missing.
- `X` If the argument `X` was a data frame or matrix of coordinates, as.matrix(`X`) is returned here. This is useful for classifying new data.

**Author(s)**

Raymaekers J., Rousseeuw P.J.

**References**


**See Also**

vcr.knn.newdata, classmap, silplot, stackedplot
vcr.neural.newdata

Examples

\[ vcrout = vcr.knn.train(iris[,1:4], iris[,5], k = 5) \]
\[ confmat.vcr(vcrout) \]
\[ stackedplot(vcrout) \]
\[ classmap(vcrout, "versicolor", classCols = 2:4) \]

# The cases misclassified as virginica are shown in blue.

# For more examples, we refer to the vignette:
\space
vignette("K_nearest_neighbors_examples")

---

vcr.neural.newdata

Prepare for visualization of a neural network classification on new data.

Description

Prepares graphical display of new data fitted by a neural net that was modeled on the training data, using the output of `vcr.neural.train` on the training data.

Usage

\[ \text{vcr.neural.newdata}(Xnew, ynew = \text{NULL}, \text{probs}, \text{vcr.neural.train.out}) \]

Arguments

- **Xnew**: data matrix of the new data, with the same number of columns as in the training data. Missing values in `Xnew` are not allowed.
- **ynew**: factor with class membership of each new case. Can be `NA` for some or all cases. If `NULL`, is assumed to be `NA` everywhere.
- **probs**: posterior probabilities obtained by running the neural net on the new data.
- **vcr.neural.train.out**: output of `vcr.neural.train` on the training data.

Value

A list with components:

- **yintnew**: number of the given class of each case. Can contain `NA`'s.
- **ynew**: given class label of each case. Can contain `NA`'s.
- **levels**: levels of the response, from `vcr.svm.train.out`.
- **predint**: predicted class number of each case. Always exists.
- **pred**: predicted label of each case.
altint number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose ynew is missing.

altlab alternative label if yintnew was given, else NA.

PAC probability of the alternative class. Is NA for cases whose ynew is missing.

fig distance of each case $i$ from each class $g$. Always exists.

farness farness of each case from its given class. Is NA for cases whose ynew is missing.

ofarness for each case $i$, its lowest $\text{fig}[i, g]$ to any class $g$. Always exists.

Author(s)
Raymaekers J., Rousseeuw P.J.

References
Raymaekers J., Rousseeuw P.J.(2021). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. (link to open access pdf)

See Also
vcr.neural.train, classmap, silplot, stackedplot

Examples

# For examples, we refer to the vignette:
vignette("Neural_net_examples")

Prepare for visualization of a neural network classification on training data.

Description
Produces output for the purpose of constructing graphical displays such as the classmap. The user first needs train a neural network. The representation of the data in a given layer (e.g. the final layer before applying the softmax function) then serves as the argument $X$ to vcr.neural.train.

Usage
vcr.neural.train($X$, $y$, $\text{probs}$, $\text{estmethod} = \text{meancov}$)
Arguments

- **X**: the coordinates of the \( n \) objects of the training data, in the layer chosen by the user. Missing values are not allowed.

- **y**: factor with the given class labels of the objects. Make sure that the levels are in the same order as used in the neural net, i.e. the columns of its binary "one-hot-encoded" response vectors.

- **probs**: posterior probabilities obtained by the neural net, e.g. in keras. For each case (row of \( X \)), the classes have probabilities that add up to 1. Each row of the matrix \( \text{probs} \) contains these probabilities. The columns of \( \text{probs} \) must be in the same order as the levels of \( y \).

- **estmethod**: function for location and covariance estimation. Should return a list with \$m\$ and \$S\$. Can be meancov (classical mean and covariance matrix) or DetMCD. If one or more classes have a singular covariance matrix, the function automatically switches to the PCA-based farness used in \texttt{vcr.svm.train}.

Value

A list with components:

- **X**: the coordinates of the \( n \) objects of the training data, in the layer chosen by the user.

- **yint**: number of the given class of each case. Can contain NA’s.

- **y**: given class label of each case. Can contain NA’s.

- **levels**: levels of \( y \)

- **predint**: predicted class number of each case. For each case this is the class with the highest posterior probability. Always exists.

- **pred**: predicted label of each case.

- **altint**: number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose \( y \) is missing.

- **altlab**: label of the alternative class. Is NA for cases whose \( y \) is missing.

- **ncolX**: number of columns in \( X \). Keep??

- **PAC**: probability of the alternative class. Is NA for cases whose \( y \) is missing.

- **computeMD**: Whether or not the farness is computed using the Mahalanobis distance.

- **classMS**: list with center and covariance matrix of each class

- **PCAfits**: if not NULL, PCA fits to each class, estimated from the training data but also useful for new data.

- **figparams**: parameters for computing fig, can be used for new data.

- **fig**: distance of each case \( i \) from each class \( g \). Always exists.

- **farness**: farness of each case from its given class. Is NA for cases whose \( y \) is missing.

- **ofarness**: for each case \( i \), its lowest \( \text{fig}[i,g] \) to any class \( g \). Always exists.
vcr.rpart.newdata

Prepare for visualization of an rpart classification on new data.

Description

Produces output for the purpose of constructing graphical displays such as the classmap on new data. Requires the output of vcr.rpart.train as an argument.

Usage

vcr.rpart.newdata(Xnew, ynew = NULL, vcr.rpart.train.out, LOO = FALSE)

Arguments

Xnew
data matrix of the new data, with the same number of columns d as in the training data. Missing values are not allowed.

ynew
factor with class membership of each new case. Can be NA for some or all cases. If NULL, is assumed to be NA everywhere.

vcr.rpart.train.out
output of vcr.rpart.train on the training data.

LOO
leave one out. Only used when testing this function on a subset of the training data. Default is LOO=FALSE.
Value

A list with components:

- **yintnew**: number of the given class of each case. Can contain NA’s.
- **ynew**: given class label of each case. Can contain NA’s.
- **levels**: levels of the response, from vcr.rpart.train.out.
- **predint**: predicted class number of each case. Always exists.
- **pred**: predicted label of each case.
- **altint**: number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose ynew is missing.
- **altlab**: alternative label if yintnew was given, else NA.
- **PAC**: probability of the alternative class. Is NA for cases whose ynew is missing.
- **fig**: distance of each case \(i\) from each class \(g\). Always exists.
- **farness**: farness of each case from its given class. Is NA for cases whose ynew is missing.
- **ofarness**: for each case \(i\), its lowest \(\text{fig}[i,g]\) to any class \(g\). Always exists.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J.(2021). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. (link to open access pdf)

See Also

vcr.rpart.train, classmap, silplot, stackedplot

Examples

library(rpart)
# load("data_titanic.rdata")
data("data_titanic")
traindata = data_titanic[which(data_titanic$dataType == "train"), -13]
str(traindata); table(traindata$y)
set.seed(123) # rpart is not deterministic
rpart.out = rpart(y ~ Pclass + Sex + SibSp +
                   Parch + Fare + Embarked,
                   data=traindata, method='class', model=TRUE)
y_train = traindata[, 12]
x_train = traindata[, -12]
mytype = list(nominal=c("Name","Sex","Ticket","Cabin","Embarked"),
              ordratio=c("Pclass"))
# These are 5 nominal columns, and one ordinal.
# The variables not listed are by default interval-scaled.

vcrtrain = vcr.rpart.train(x_train, y_train, rpart.out, mytype)

testdata = data_titanic[which(data_titanic$dataType == "test"), -13]

dim(testdata)

x_test = testdata[, -12]
y_test = testdata[, 12]

vcrtest = vcr.rpart.newdata(x_test, y_test, vcrtrain)

cnfmat.vcr(vcrtest)

silplot(vcrtest, classCols = c(2,4))

classmap(vcrtest, "casualty", classCols = c(2,4))

classmap(vcrtest, "survived", classCols = c(2,4))

---

vcr.rpart.train  Prepare for visualization of an rpart classification on training data.

---

Description

Produces output for the purpose of constructing graphical displays such as the classmap. The user first needs to train a classification tree on the data by rpart::rpart. This then serves as an argument to vcr.rpart.train.

Usage

vcr.rpart.train(X, y, trainfit, type = list(),
    k = 5, stand = TRUE)

Arguments

X  A rectangular matrix or data frame, where the columns (variables) may be of mixed type and may contain NA’s.

y  factor with the given class labels. It is crucial that X and y are exactly the same as in the call to rpart::rpart. y is allowed to contain NA’s.

k  the number of nearest neighbors used in the farness computation.

trainfit  the output of an rpart::rpart training cycle.

type  list for specifying some (or all) of the types of the variables (columns) in X, used for computing the dissimilarity matrix, as in cluster::daisy. The list may contain the following components: "ordratio" (ratio scaled variables to be treated as ordinal variables), "logratio" (ratio scaled variables that must be logarithmically transformed), "asymm" (asymmetric binary) and "symm" (symmetric binary variables). Each component’s value is a vector, containing the names or the numbers of the corresponding columns of X. Variables not mentioned in the type list are interpreted as usual (see argument X).

stand  whether or not to standardize numerical (interval scaled) variables by their range as in the original cluster::daisy code for the farness computation. Defaults to TRUE.
Value

A list with components:

- **X**: The input data $X$. Keep??
- **yint**: number of the given class of each case. Can contain NA's.
- **y**: given class label of each case. Can contain NA's.
- **levels**: levels of $y$
- **predint**: predicted class number of each case. For each case this is the class with the highest posterior probability. Always exists.
- **pred**: predicted label of each case.
- **altint**: number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose $y$ is missing.
- **altlab**: label of the alternative class. Is NA for cases whose $y$ is missing.
- **PAC**: probability of the alternative class. Is NA for cases whose $y$ is missing.
- **figparams**: parameters for computing fig, can be used for new data.
- **fig**: distance of each case $i$ from each class $g$. Always exists.
- **farness**: farness of each case from its given class. Is NA for cases whose $y$ is missing.
- **ofarness**: for each case $i$, its lowest $fig[i, g]$ to any class $g$. Always exists.
- **trainfit**: the trainfit used to build the VCR object.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J.(2021). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. [link to open access pdf](link)

See Also

- [vcr.rpart.newdata](#), [classmap](#), [silplot](#), [stackedplot](#)

Examples

```r
library(rpart)
# load("data_titanic.rdata")
data("data_titanic")
traindata = data_titanic[which(data_titanic$dataType == "train"), -13]
str(traindata); table(traindata$y)
set.seed(123) # rpart is not deterministic
rpart.out = rpart(y ~ Pclass + Sex + SibSp +
   Parch + Fare + Embarked,
   data=traindata, method='class', model=TRUE)
```
y_train = traindata[, 12]
x_train = traindata[,-12]
mytype = list(nominal=c("Name","Sex","Ticket","Cabin","Embarked"),
             ordratio=c("Pclass"))
# These are 5 nominal columns, and one ordinal.
# The variables not listed are by default interval-scaled.
vcrtrain = vcr.rpart.train(x_train, y_train, rpart.out, mytype)
confmat(vcrtrain)
silplot(vcrtrain, classCols = c(2,4))
# Quasi residual plot:
xm = x_train$Age[which(x_train$Sex == "male")]
ym = vcrtrain$PAC[which(x_train$Sex == "male")]
plot(xm, ym, main = "quasi residual plot for males",
     xlab = "Age (years)", ylab = "P[alternative class]"
)
lom = loess(ym ~ xm)
lines(0:82, predict(lom, 0:82), col="red", lwd = 2)
text(x=17,y=0.56,"loess curve",col="red", cex = 1)
classmap(vcrtrain, "casualty", classCols = c(2,4))
classmap(vcrtrain, "survived", classCols = c(2,4))

---

vcr.svm.newdata

Prepare for visualization of a support vector machine classification on new data.

Description

Carries out a support vector machine classification of new data using the output of vcr.svm.train on the training data, and computes the quantities needed for its visualization.

Usage

vcr.svm.newdata(Xnew, ynew=NULL, vcr.svm.train.out)

Arguments

Xnew  
data matrix of the new data, with the same number of columns as in the training data. Missing values in Xnew are not allowed.

ynew  
factor with class membership of each new case. Can be NA for some or all cases. If NULL, is assumed to be NA everywhere.

vcr.svm.train.out  
output of vcr.svm.train on the training data.

Value

A list with components:

yintnew  
number of the given class of each case. Can contain NA’s.

ynew  
given class label of each case. Can contain NA’s.
levels levels of the response, from `vcr.svm.train.out`.
predint predicted class number of each case. Always exists.
pred predicted label of each case.
altint number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose `ynew` is missing.
altlab alternative label if `yintnew` was given, else NA.
PAC probability of the alternative class. Is NA for cases whose `ynew` is missing.
fig distance of each case $i$ from each class $g$. Always exists.
farness farness of each case from its given class. Is NA for cases whose `ynew` is missing.
ofarness for each case $i$, its lowest fig[$i$, $g$] to any class $g$. Always exists.

Author(s)
Raymaekers J., Rousseeuw P.J.

References

See Also
`vcr.svm.train`, `classmap`, `silplot`, `stackedplot`, `e1071::svm`

Examples

```r
library(e1071)
set.seed(1); X = matrix(rnorm(200*2),ncol=2)
X[1:100,] = X[1:100,]+2
X[101:150,] = X[101:150,-2]
y = as.factor(c(rep("blue",150),rep("red",50)))
# We now fit an SVM with radial basis kernel to the data:
set.seed(1) # to make the result of svm() reproducible.
svmfit = svm(y~.,data=data.frame(X=X,y=y),scale=FALSE,kernel="radial",
            cost=10, gamma=1, probability=TRUE)
vcr.train = vcr.svm.train(X, y, svfit=svmfit)
# As "new" data we take a subset of the training data:
inds = c(1:25,101:125,151:175)
vcr.test = vcr.svm.newdata(X[inds,],y[inds],vcr.train)
plot(vcr.test$PAC,vcr.train$PAC[inds]); abline(0,1) # match
plot(vcr.test$farness,vcr.train$farness[inds]); abline(0,1)
confmat.vcr(vcr.test)
cols = c("deepskyblue3","red")
stackedplot(vcr.test, classCols = cols)
classmap(vcr.train, "blue", classCols = cols) # for comparison
classmap(vcr.test, "blue", classCols = cols)
classmap(vcr.train, "red", classCols = cols) # for comparison
```
vcr.svm.train

Prepare for visualization of a support vector machine classification on training data.

Description

Produces output for the purpose of constructing graphical displays such as the classmap. The user first needs to run a support vector machine classification on the data by e1071::svm, with the option probability = TRUE. This classification can be with two or more classes. The output of e1071::svm is then an argument to vcr.svm.train. As e1071::svm does not output the data itself, it needs to be given as well, in the arguments X and y.

Usage

vcr.svm.train(X, y, svfit, ortho = FALSE)

Arguments

X matrix of data coordinates, as used in e1071::svm. Missing values are not allowed.
y factor with the given (observed) class labels. It is crucial that X and y are exactly the same as in the call to e1071::svm.
svfit an object returned by e1071::svm, called with exactly the same X and y as above.
ortho If TRUE, will compute farness in the orthogonal complement of the vector beta given by e1071::svm. Is only possible for 2 classes, else there would be several beta vectors.

Value

A list with components:

yint number of the given class of each case. Can contain NA's.
y given class label of each case. Can contain NA's.
levels levels of the response y.
predint predicted class number of each case. Always exists.
pred predicted label of each case.
altint number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose y is missing.

altlab label of the alternative class. Is NA for cases whose y is missing.
PAC probability of the alternative class. Is NA for cases whose y is missing.
figparams parameters used in fig, can be used for new data.
fig distance of each case i from each class g. Always exists.
farness farness of each case from its given class. Is NA for cases whose y is missing.
ofarness for each case i, its lowest fig[i,g] to any class g. Always exists.
svfit as it was input, will be useful for new data.
X the matrix of data coordinates from the arguments. This is useful for classifying new data.

Author(s)
Raymaekers J., Rousseeuw P.J.

References

See Also
vcr.knn.newdata, classmap, silplot, stackedplot, e1071::svm

Examples
```r
library(e1071)
set.seed(1); X = matrix(rnorm(200*2),ncol=2)
X[1:100,] = X[1:100,]+2
X[101:150,] = X[101:150,-]2
y = as.factor(c(rep("blue",150),rep("red",50)))
cols = c("deepskyblue3","red")
plot(X,col=cols[as.numeric(y)],pch=19)
# We now fit an SVM with radial basis kernel to the data:
set.seed(1) # to make the result of svm() reproducible.
svmfit = svm(y~.,data=data.frame(X=X,y=y),scale=FALSE,kernel="radial",
  cost=10, gamma=1, probability=TRUE)
plot(svmfit$decision.values,col=cols[as.numeric(y)]); abline(h=0)
# so the decision values separate the classes reasonably well.
plot(svmfit,data=data.frame(X=X,y=y),X.2~X.1,col=cols)
# The boundary is far from linear (but in feature space it is).
vcr.train = vcr.svm.train(X, y, svfit=svmfit)
confmat.vcr(vcr.train)
stackedplot(vcr.train, classCols = cols)
classmap(vcr.train, "blue", classCols = cols)
classmap(vcr.train, "red", classCols = cols)
```
# For more examples, we refer to the vignettes:
vignette("Support_vector_machine_examples")
Index

classmap, 2, 16–21, 23, 25, 26, 28, 30–33, 35–37
cluster::daisy, 22, 25, 32
confmat.vcr, 4
data_bookReviews, 6
data_floralbuds, 7
data_instagram, 8
data_titanic, 9
e1071::svm, 12, 35–37
graphics::identify, 3
graphics::plot, 3
kernlab::kernelMatrix, 10
makeFV, 10, 10, 11, 12
makeKernel, 10, 11, 12
randomForest::randomForest, 21, 22
robustbase::covMcd, 18
rpart::rpart, 32
silplot, 13, 17, 19, 21, 23, 25, 26, 28, 30, 31, 33, 35, 37
stackedplot, 14, 17, 19, 21, 23, 25, 26, 28, 30, 31, 33, 35, 37
stats::dist, 25
vcr.da.newdata, 4, 5, 14, 15, 16, 19
vcr.da.train, 4, 5, 14–17, 18
vcr.forest.newdata, 4, 5, 14, 15, 20, 23
vcr.forest.train, 4, 5, 14, 15, 20, 21, 21
vcr.knn.newdata, 4, 5, 14, 15, 23, 26, 37
vcr.knn.train, 4, 5, 14, 15, 23–25, 25
vcr.neural.newdata, 4, 5, 14, 15, 27, 30
vcr.neural.train, 4, 5, 14, 15, 27, 28, 28
vcr.rpart.newdata, 4, 5, 14, 15, 30, 33
vcr.rpart.train, 4, 5, 14, 15, 30–32, 32
vcr.svm.newdata, 4, 5, 14, 15, 34
vcr.svm.train, 4, 5, 14, 15, 29, 34–36, 36