Package ‘SVMMaj’

January 26, 2019

Type Package
Title Implementation of the SVM-Maj Algorithm
Version 0.2.9
Date 2019-01-26
Author Hoksan Yip, Patrick J.F. Groenen, Georgi Nalbantov
Maintainer Hok San Yip <hoksan@gmail.com>
Description Implements the SVM-Maj algorithm to train data with support vector machine as described in Groenen et al. (2008) <doi:10.1007/s11634-008-0020-9>. This algorithm uses two efficient updates, one for linear kernel and one for the nonlinear kernel.
Imports reshape2, scales, gridExtra, dplyr, ggplot2, kernlab
Depends R (>= 2.13.0), stats, graphics
Suggests utils, testthat, magrittr, xtable
License GPL-2
LazyData Yes
RoxygenNote 6.1.1
VignetteBuilder utils
NeedsCompilation no
Repository CRAN
Date/Publication 2019-01-26 13:10:03 UTC

R topics documented:

  auc .................................................. 2
  AusCredit ......................................... 3
  classification ..................................... 4
  diabetes ........................................... 4
  getHinge .......................................... 5
  isb ................................................ 6
  isplinebasis ...................................... 7
Returns the area under the curve value

**Usage**

```r
auc(q, y = attr(q, "y"))
```

**Arguments**

- `q`: the predicted values
- `y`: a list of the actual classes of `q`

**Value**

the area under the curve value

**Examples**

```r
df <- with(diabetes, cbind(y, X))
lm.y <- glm(y ~ ., data = df, family = binomial())
print(with(lm.y, auc(fitted.values, y)))
```
AusCredit  

Australian Credit Approval Dataset  

Description

This file concerns credit card applications of 690 households.

Format

This data set has been split into two components for the convenience of the model training.

data.frame object \( X \) consists of 6 numerical and 8 categorical attributes. The labels have been changed for the convenience of the statistical algorithms. For example, attribute 4 originally had 3 labels p,g,gg and these have been changed to labels 1,2,3.

Factor \( y \) indicates whether the application has been Accepted or Rejected

The training set \( \text{AusCredit.tr} \) contains a randomly selected set of 400 subjects, and \( \text{AusCredit.te} \) contains the remaining 290 subjects. \( \text{AusCredit} \) contains all 690 objects.

Details

All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data.

This dataset is interesting because there is a good mix of attributes – continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values.

Source


Examples

```
attach(AusCredit)
summary(X)
summary(y)
detach(AusCredit)
```
classification

Show the classification performance

Description

Given the predicted value q and the observed classes y, it shows an overview of the prediction performances with hit rates, misclassification rates, true positives (TP), false positives (FP) and precision.

Usage

classification(q, y, classes = c("1", "0"), weights = NULL)

Arguments

q the predicted values
y a list of the actual classes of q
classes a character vector with the labels of the two classes
weights an optional parameter to specify a weighted hit rate and misclassification rate

Value

a list with three elements, matrix equals the confusion matrix, overall equals the overall prediction performance and in measures the measures per class is stored.

diabetes

Pima Indians Diabetes Data Set

Description

From National Institute of Diabetes and Digestive and Kidney Diseases.

Format

X is a data frame of 768 female patients with 8 attributes.

no.pregnant number of pregnancies.
glucose plasma glucose concentration in an oral glucose tolerance test
blood.press diastolic blood pressure (mm Hg)
triceps.thick triceps skin fold thickness (mm)
insulin 2-Hour serum insulin (mu U/ml)
BMI body mass index (weight in kg/(height in m)^2)
pedigree diabetes pedigree function
age age in years
getHinge

y contains the class labels: Yes or No, for diabetic according to WHO criteria. The training set diabetes.tr contains a randomly selected set of 600 subjects, and diabetes.te contains the remaining 168 subjects. diabetes contains all 768 objects.

Details

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Source


References


Examples

attach(diabetes)
summary(x)
summary(y)

---

getHinge

Hinge error function of SVM-Maj

Description

This function creates a function to compute the hinge error, given its predicted value q and its class y, according to the loss term of the Support Vector machine loss function.

Usage

gethinge(hinge = "quadratic", delta = 3, eps = 1e-08)

Arguments

- **hinge**: Hinge error function to be used, possible values are 'absolute', 'quadratic' and 'huber'
- **delta**: The parameter of the huber hinge (only if hinge = 'huber').
- **eps**: Specifies the maximum steepness of the quadratic majorization function

\[ m(q) = a \cdot q^2 - 2 \cdot b \cdot q + c \]

where \( a \leq 0.25 \cdot \text{eps}^{-1} \).
Value

The hinge error function with arguments \( q \) and \( y \) to compute the hinge error. The function returns a list with the parameters of the majorization function SVM-Maj \((a, b, c)\) and the loss error of each object \((\text{loss})\).

References


See Also

svmmaj

Examples

```r
hingefunction <- getHinge()
# plot hinge function value and, if specified, 
# the majorization function at \( z \)
# plot(hingefunction, \( z = 3 \))
# generate loss function value
loss <- hingefunction(q = -10:10, y = 1)$loss
print(loss)
plot(hingefunction, \( z = 3 \))
```

---

**isb**

*I-spline basis of each column of a given matrix*

Description

Create a I-spline basis for an array. \( \text{isb} \) will equally distribute the knots over the value range using quantiles.

Usage

```r
isb(x, spline.knots = 0, knots = NULL, spline.degree = 1)
```

Arguments

- **x**: The predictor variable, which will be transformed into I-spline basis.
- **spline.knots**: Number of inner knots to use. \( \text{isb} \) will equally distribute the knots over the value range using quantiles. \( \text{spline.knots} \) will only be used if \( \text{knots} \) is not given.
- **knots**: An array consisting all knots (boundary knots as well as the interior knots) to be used to create the spline basis.
- **spline.degree**: The polynomial degree of the spline basis.
isplinebasis

Value

The I-spline with the used spline settings as attribute. The spline settings attribute can transform the same attribute of any other objects using the same knots.

Author(s)

Hok San Yip, Patrick J.F. Groenen, Georgi Nalbantov

References


See Also

svmmaj

Examples

```r
## plot the spline transformation given a monotone sequence
B0 <- isb(0:100, spline.knots = 2, spline.degree = 3)
plot(NULL, xlim = c(0, 140), ylim = c(0, 1), xlab = 'x', ylab = 'I-spline')
for(i in 1:ncol(B0))
  lines(B0[,i], col = i, lwd = 3)
legend('bottomright', legend = 1:ncol(B0), col = 1:ncol(B0),
  lty = 1, lwd = 3, title = 'Spline Columns')
## create I-spline basis for the first 50 observations
x <- iris$Sepal.Length
B1 <- isb(x[1:50], spline.knots = 4, spline.degree = 3)
## extracting the spline transformation settings
spline.param <- attr(B1, 'splineinterval')
## use the same settings to apply to the next 50 observations
B2 <- isb(x[-(1:50)], spline.degree = 3, knots = spline.param)
```

---

isplinebasis

*Transform a given data into I-splines*

Description

Inner function call to create I-splines based on the user defined knots and polynomial degree d of the splines

Usage

```r
isplinebasis(x, knots, d)
```
**Arguments**

- `x`: a scalar or vector of values which will be transformed into splines
- `knots`: a vector of knot values of the splines
- `d`: the polynomial degree of the splines

**Value**

A matrix with for each value of `x` the corresponding spline values.

---

**normalize**

*Normalize/standardize the columns of a matrix*

**Description**

Standardize the columns of an attribute matrix `X` to zscores, to the range `[0 1]` or a prespecified scale.

**Usage**

`normalize(x, standardize = "zscore")`

**Arguments**

- `x`: An attribute variable which will be scaled.
- `standardize`: Either a string value denoting a predefined scaling, or a list with values `a` and `b` corresponding with the numeric centering and scaling, that is, using the function `x * standardize$b -standardize$a`.

**Value**

The standardized matrix. The numeric centering and scalings used are returned as attribute "standardize".

**Author(s)**

Hok San Yip, Patrick J.F. Groenen, Georgi Nalbantov

**References**


**See Also**

`svmmaj`
plot.hinge

Examples

```r
## standardize the first 50 objects to zscores
x <- iris$Sepal.Length
x1 <- normalize(x[1:50], standardize = 'zscore')
## use the same settings to apply to the next 100 observations
x2 <- normalize(x[-(1:50)], standardize = attr(x1, 'standardization'))
```

Description

This function plots the hinge object created by `getHinge`.

Usage

```r
## S3 method for class 'hinge'
plot(x, y = 1, z = NULL, ...)
```

Arguments

- `x`: The hinge object returned from `getHinge`.
- `y`: Specifies the class (-1 or 1) to be plotted for the hinge error.
- `z`: If specified, the majorization function with the supporting point `z` will also be plotted.
- `...`: Other arguments passed to plot method.

Examples

```r
hingefunction <- getHinge()
## plot hinge function value
plot(hingefunction, z = 3)
```

plot.svvmajcrossval

Plot the cross validation output

Description

Shows the results of the cross validation graphically. Possible graphics are among others the distribution of the predicted values per class per lambda value and the misclassification rate per lambda.
Usage

```r
## S3 method for class 'svmmajcrossval'
plot(x, type = "grid", ...)
```

Arguments

- `x`: the `svmmajcrossval` object
- `type`: the type of graph being shown, possible values are 'grid' for the misclassification rate per lambda value, 'profile' the distribution of predicted values of the classes per lambda value
- `...`: Further arguments passed to or from other methods.

---

`plotWeights` | *Plot the weights of all attributes from the trained SVM model*

### Description

Shows, one graph per attribute, the weights of all attributes. The type of graph depends on the type of the attribute: the spline line of the corresponding attribute in case a spline has been used, a bar plot for categorical and logical values, and a linear line for all other type of the attribute values. This function cannot be used in a model with a non-linear kernel.

#### Usage

```r
plotWeights(object, plotdim = c(3, 3), ...)
```

Arguments

- `object`: The model returned from `svmmaj`.
- `plotdim`: A vector of the form `c(nr, nc)`. Subsequent figures will be drawn in an `nr`-by-`nc` array on the device.
- `...`: other parameters given to the `plot` function

---

`predict.svmmaj` | *Out-of-Sample Prediction from Unseen Data.*

### Description

This function predicts the predicted value (including intercept), given a previous trained model which has been returned by `svmmaj`. 
predict.svmmaj

Usage
### S3 method for class 'svmmaj'
predict(object, X.new, y = NULL, weights = NULL,
       show.plot = FALSE, ...)

Arguments

- **object**: Model which has been trained beforehand using `svmmaj`.
- **X.new**: Attribute matrix of the objects to be predicted, which has the same number of attributes as the untransformed attribute matrix in `model`.
- **y**: The actual class labels (only if `show.plot` is `TRUE`).
- **weights**: The weight of observation as the relative importance of the prediction error of the observation.
- **show.plot**: If `show.plot=TRUE`, it plots the density of the predicted value for both class labels, if `y` is not specified, the density of all objects will be plotted.
- **...**: Arguments to be passed to methods.

Value

The predicted value (including intercept) of class `q.svmmaj`, with attributes:

- **y**: The observed class labels of each object.
- **yhat**: The predicted class labels of each object.
- **classes**: The class labels.

Author(s)

Hok San Yip, Patrick J.F. Groenen, Georgi Nalbantov

References


See Also

`svmmaj`

Examples

```r
attach(AusCredit)

## model training
model <- svmmaj(X[1:400,], y[1:400], hinge = 'quadratic', lambda = 1)
## model prediction
q4 <- predict(model, X[-(1:400),], y[-(1:400)], show.plot = TRUE)
q4
```
performData

Perform the transformation based on predefined settings

Description

Given the input parameters, which are generated from transformData, it performs the same transformation with the same settings to the given input.

Usage

```r
## S3 method for class 'transDat'
predict(x, attrib = NULL, values = NULL,
        standardization = NULL, splineInterval = NULL, splineDegree = NULL)```

Arguments

- `x`: a (new) vector of numerics to be transformed
- `attrib`: either a list of settings, or `NULL` in case the attributes are given as separate input
- `values`: a vector of levels in case `x` is a factor
- `standardization`: the standardization rules from normalize
- `splineInterval`: the knots to be used for spline basis
- `splineDegree`: the polynomial degree of the splines

Value

A transformed data based on the user defined settings.

SVM-Maj Algorithm

Description

SVM-Maj is an algorithm to compute a support vector machine (SVM) solution. In its most simple form, it aims at finding hyperplane that optimally separates two given classes. This objective is equivalent to finding a linear combination of `k` predictor variables to predict the two classes for `n` observations. SVM-Maj minimizes the standard support vector machine (SVM) loss function. The algorithm uses three efficient updates for three different situations: primal method which is efficient in the case of `n > k`, the decomposition method, used when the matrix of predictor variables is not of full rank, and a dual method, that is efficient when `n < k`. Apart from the standard absolute hinge error, SVM-Maj can also handle the quadratic and the Huber hinge.
Usage

```r
## S3 method for class 'q.svmmaj'
print(x, ...)

svmmaj(X, y, lambda = 1, weights.obs = 1, weights.var = 1,
       scale = c("interval", "zscore", "none"),
       spline.knots = 0,
       spline.degree = 1L, kernel = vanilladot, kernel.sigma = 1,
       kernel.scale = 1, kernel.degree = 1, kernel.offset = 1,
       hinge = c("absolute", "quadratic", "huber", "logitistic"),
       hinge.delta = 1e-08, options = setSVMOptions(),
       initial.point = NULL, verbose = FALSE, na.action = na.omit, ...)

## Default S3 method:
svmmaj(X, y, lambda = 1, weights.obs = 1,
       weights.var = 1, scale = c("interval", "zscore", "none"),
       spline.knots = 0, spline.degree = 1L, kernel = vanilladot,
       kernel.sigma = 1, kernel.scale = 1, kernel.degree = 1,
       kernel.offset = 1, hinge = c("absolute", "quadratic", "huber",
                                  "logitistic"), hinge.delta = 1e-08, options = setSVMOptions(),
       initial.point = NULL, verbose = FALSE, na.action = na.omit, ...)
```

Arguments

- **x**: the svmmaj object as result of `svmmaj`
- **...**: Other arguments passed to methods.
- **X**: A data frame (or object coercible by `as.data.frame` to a data frame) consisting the attributes, the class of each attribute can be either numeric, logical or factor.
- **y**: A factor (or object coercible by `factor` to a factor) consisting the class labels.
- **lambda**: Regularization parameter of the penalty term.
- **weights.obs**: a vector of length \( n \) with the nonnegative weight for the residual of each object (with length \( n \)). If the length is 2, then it specifies the weight per class.
- **weights.var**: a vector of length \( k \) with weights for each attribute.
- **scale**: Specifies whether the columns of attribute matrix \( X \) needs to be standardized into zscores or to the interval \([0 \ 1]\). Possible values are: none, zscore and interval. Moreover, the standardization parameters can be given instead.
- **spline.knots**: equals the number of internal knots of the spline basis. When the number of knots exceeds the number of (categorical) values of an explanatory variable, the duplicate knots will be removed using `unique`. For no splines, use `spline.knots = 0`.
- **spline.degree**: equals the polynomial degree of the splines, for no splines: `spline.degree = 1`.
- **kernel**: Specifies which kernel function to be used (see `dots` of package `kernlab`). Default kernel is the linear kernel.
- **kernel.sigma**: additional parameters used for the kernel function (see `dots`)
- **kernel.scale**: additional parameters used for the kernel function (see `dots`)
kernel.degree  additional parameters used for the kernel function (see dots)
kernell.offset  additional parameters used for the kernel function (see dots)
hinge         Specifies with hinge function from getHinge should be used.
hinge.delta   The parameter of the huber hinge (only if hinge = 'huber').
options       additional settings used in the svmmaj algorithm
initial.point Initial solution.
verbose       TRUE shows the progress of the iteration.
na.action     Generic function for handling NA values.

Details

The following settings can be added as element in the options parameter: decomposition Specifies whether the QR decomposition should be used for efficient updates. Possible values are 'svd' for Singular value decomposition (Eigenvalue decomposition for non-linear kernel) or 'chol' for Cholesky (or QR decomposition in case of linear kernel).

convergence Specifies the convergence criterion of the algorithm. Default is 1e-08.

increase.step The iteration number from which relaxed update will be used. eps The relaxation of the majorization function for absolute hinge: .25 * eps^-1 is the maximum steepness of the majorization function.

check.positive Specifies whether a check has to be made for positive input values. max.iter maximum number of iterations to use

Value

Returns a svmmaj-class object, of which the methods plot, plotWeights, summary and predict can be applied. (see also predict.svmmaj and print.svmmaj)

Author(s)

Hok San Yip, Patrick J.F. Groenen, Georgi Nalbantov

References


See Also

dots for the computations of the kernels. predict.svmmaj normalize isb getHinge

Examples

```r
## using default settings
model1 <- svmmaj(
    diabetes$x, diabetes$y, hinge = 'quadratic', lambda = 1)
summary(model1)
```
weights.obs = list(positive = 2, negative = 1)
## using radial basis kernel
library(kernlab)
model2 <- svmmaj(
  diabetes$x, diabetes$y, hinge = 'quadratic', lambda = 1,
  weights.obs = weights.obs, scale = 'interval',
  kernel = rbf,  
  kernel.sigma = 1
)
summary(model2)
## I-spline basis
library(ggplot2)
model3 <- svmmaj(
  diabetes$x, diabetes$y, weight.obs = weight.obs,
  spline.knots = 3, spline.degree = 2
)
plotWeights(model3, plotdim = c(2, 4))

---

**print.svmmaj**

**Print Svmmaj class**

**Description**

Trained SVM model as output from svmmaj. The returning object consist of the following values:

- **call** The function specifications which has been called.
- **lambda** The regularization parameter of the penalty term which has been used.
- **loss** The corresponding loss function value of the final solution.
- **iteration** Number of iterations needed to evaluate the algorithm.
- **X** The attribute matrix of \( \text{dim}(X) = c(n,k) \).
- **y** The vector of length \( n \) with the actual class labels. These labels can be numeric \([0,1]\) or two strings.
- **classes** A vector of length \( n \) with the predicted class labels of each object, derived from \( \text{q.tilde} \).
- **Xtrans** The attribute matrix \( X \) after standardization and (if specified) spline transformation.
- **norm.param** The applied normalization parameters (see normalize).
- **splineInterval** The spline knots which has been used (see isb).
- **splineLength** Denotes the number of spline basis of each explanatory variable in \( X \).
- **method** The decomposition matrices used in estimating the model.
- **hinge** The hinge function which has been used (see getHinge).
- **beta** If identified, the beta parameters for the linear combination (only available for linear kernel).
- **q** A vector of length \( n \) with predicted values of each object including the intercept.
- **nSV** Number of support vectors.
Usage

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

## S3 method for class 'svmmaj'
summary(object, ...)

## S3 method for class 'summary.svmmaj'
print(x, ...)

## S3 method for class 'svmmaj'
plot(x, ...)
```

Arguments

- `x` the `svmmaj` object as result of `svmmaj`
- `...` further arguments passed to or from other methods.
- `object` the `svmmaj` object as result of `svmmaj`

---

**Description**

Prints the result from the cross validation procedure in `svmmajcrossval`.

Usage

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

## S3 method for class 'svmmajcrossval'
summary(object, ...)
```

Arguments

- `x` the cross-validation output from `svmmajcrossval`
- `...` ignored
- `object` the output object from `svmmajcrossval`
**roccurve**  
*Plot the ROC curve of the predicted values*

**Description**

Given the predicted values \( q \) and its corresponding observed classes \( y \), it shows its separation performances by showing the roc-curve.

**Usage**

```r
roccurve(q, y = attr(q, "y"), class = 1, ...)
```

**Arguments**

- \( q \)  the predicted values
- \( y \)  a list of the actual classes of \( q \)
- \( \text{class} \)  the base class to show the roc-curve
- \( ... \)  additional parameters given as input to the `plot` function

**Examples**

```r
model <- svm(diabetes$X, diabetes$y)
roccurve(model$q)
```

---

**supermarket1996**  
*Supermarket data 1996*

**Description**

This

**Format**

This dataframe contains the following columns

- StoreIdentifier of the store
- CITYThe city of the store
- ZIPThe zip code of the store
- GROCERY_sum
- GROCOCOUP_sum
- AGE9
- AGE60
- ETHNIC
• EDUC
• NOCAR
• INCOME
• INCSIGMA
• HSIZEAVG
• HSIZE1
• HSIZE2
• HSIZE34
• HSIZE567
• HH3PLUS
• HH4PLUS
• HHSINGLE
• HHLARGE
• WORKWOM
• SINHOUSE
• DENSITY
• HVAL150
• HVAL200
• HVALMEAN
• SINGLE
• RETIRED
• UNEMP
• WRKCH5
• WRKCH17
• NWRKCH5
• NWRKCH17
• WRKCH
• NWRKCH
• WRKWCH
• WRKWNCH
• TELEPHN
• MORTGAGE
• NWHITE
• POVERTY
• SHPCONS
• SHPHURR
• SHPAVID
• SHPKSTR
• SHPUNFT
• SHPBIRD
• SHOPINDEX
• SHPINDEX
Examples

head(supermarket1996, 3)

svmmajcrossval  \textit{k-fold Cross-Validation of SVM-Maj}

Description

This function performs a gridsearch of k-fold cross-validations using SVM-Maj and returns the combination of input values which has the best forecasting performance.

Usage

svmmajcrossval(X, y, search.grid = list(lambda = 2^seq(-5, -5, length.out = 19)), ..., convergence = 1e-04, weights.obs = 1, check.positive = TRUE, mc.cores = getOption("mc.cores"), options = NULL, verbose = FALSE, ngroup = 5, groups = NULL, return.model = FALSE)

Arguments

\begin{itemize}
  \item \texttt{X} \hspace{2cm} A data frame (or object coercible by \texttt{as.data.frame} to a data frame) consisting the attributes.
  \item \texttt{y} \hspace{2cm} A factor (or object coercible by \texttt{factor} to a factor) consisting the class labels.
  \item \texttt{search.grid} \hspace{2cm} A list with for each factor the range of values to search for.
  \item \texttt{...} \hspace{2cm} Other arguments to be passed through \texttt{svmmaj}.
  \item \texttt{convergence} \hspace{2cm} Specifies the convergence criterion for \texttt{svmmaj}. Default is $1e^{-08}$.
  \item \texttt{weights.obs} \hspace{2cm} Weights for the classes.
  \item \texttt{check.positive} \hspace{2cm} Specifies whether a check should be performed for positive \texttt{lambda} and \texttt{weights.obs}.
  \item \texttt{mc.cores} \hspace{2cm} the number of cores to be used (for parallel computing)
  \item \texttt{options} \hspace{2cm} additional settings used in the \texttt{svmmaj} algorithm
  \item \texttt{verbose} \hspace{2cm} =TRUE shows the progress of the cross-validation.
  \item \texttt{ngroup} \hspace{2cm} The number of groups to be divided into.
  \item \texttt{groups} \hspace{2cm} A predetermined group division for performing the cross validation.
  \item \texttt{return.model} \hspace{2cm} =TRUE estimates the model with the optimal parameters.
\end{itemize}
Value

loss.opt The minimum (weighted) missclassification rate found in out-of-sample training along the search grid.

param.opt The level of the factors which gives the minimum loss term value.

loss.grp A list of missclassification rates per hold-out sample groups

A vector defining the cross-validation groups which has been used.

qhat The estimated out-of-sample predicted values in the cross-validation.

qhat.in The trained predicted values

param.grid The matrix of all gridpoints which has been performed during the cross-validation, with its corresponding weighted out-of-sample missclassification rate.

model The svmmaj-object with the estimated model using the optimal parameters found in the cross-validation.

Author(s)

Hok San Yip, Patrick J.F. Groenen, Georgi Nalbantov

References


See Also

svmmaj

Examples

Xt <- diabetes$X
yt <- diabetes$y

## performing gridsearch with k-fold cross-validation
results <- svmmajcrossval(
  Xt, yt, scale = 'interval',
  mc.cores = 2,
  ngroup = 5,
  return.model = TRUE
)

summary(results$model)
results
plot(results)
plot(results, 'profile')
transformdata

Transform the data with normalization and/or spline basis

Description
Performs subsequently a normalization of the input data and creating spline basis based on the user defined input.

Usage

\[
\text{transformdata}(x, \ \text{standardize} = \text{c}("\text{interval}", "\text{zscore}", "\text{none}"), \\
\text{spline.knots} = 0, \text{spline.degree} = 1)
\]

Arguments

- \text{x} a single column of values as input for the data transformation
- \text{standardize} Either a string value denoting a predefined scaling, or a list with values \text{a} and \text{b} corresponding with the numeric centering and scaling, that is, using the function \text{x} * \text{standardize$b - standardize$a}.
- \text{spline.knots} Number of inner knots to use. \text{isb} will equally distribute the knots over the value range using quantiles. \text{spline.knots} will only be used if \text{knots} is not given.
- \text{spline.degree} The polynomial degree of the spline basis.

Value
transformed data in spline basis or (in case of no spline) a normalized vector

voting

Congressional Voting Records Data Set

Description
1984 United Stated Congressional Voting Records; Classify as Republican or Democrat.

Format
\text{X} is a data frame with 434 congress members and 16 attributes: 16 key votes identified by the Congressional Quarterly Almanac (CQA). All attributes are binary values, with 1= yes and 0= no.

\begin{align*}
\text{X1} & \text{ handicapped-infants} \\
\text{X2} & \text{ water-project-cost-sharing} \\
\text{X3} & \text{ adoption-of-the-budget-resolution} \\
\text{X4} & \text{ physician-fee-freeze} \\
\text{X5} & \text{ el-salvador-aid}
\end{align*}
X6  religious-groups-in-schools
X7  anti-satellite-test-ban
X8  aid-to-nicaraguan-contras
X9  mx-missile
X10  immigration
X11  synfuels-corporation-cutback
X12  education-spending
X13  superfund-right-to-sue
X14  crime
X15  duty-free-exports
X16  export-administration-act-south-afric

y consists factors which denotes whether the congress member is a Republican or a Democrat.

The training set voting.tr contains a randomly selected set of 300 subjects, and voting.te contains the remaining 134 subjects. voting contains all 434 objects.

Details

This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition).

Source


Examples

attach(voting)
summary(X)
summary(y)

X.svmmaj  Returns transformed attributes

Description

For efficiency use in svmmajcrossval

Usage

X.svmmaj(object, X.new, weights = NULL)
Arguments

- **object**: Model which has been trained beforehand using `svmmaj`.
- **X.new**: Attribute matrix of the objects to be predicted, which has the same number of attributes as the untransformed attribute matrix in `model`.
- **weights**: The weight of observation as the relative importance of the prediction error of the observation.
Index

*Topic datasets
  AusCredit, 3
diabetes, 4
  supermarket1996, 17
  voting, 21

as.data.frame, 13, 19
auc, 2
  AusCredit, 3
classification, 4
diabetes, 4
dots, 13, 14

factor, 13, 19
getHinge, 5, 14, 15

isb, 6, 14, 15
isplinebasis, 7
normalize, 8, 14, 15

plot.hinge, 9
plot.svmmaj(print.svmmaj), 15
plot.svmmajcrossval, 9
plotWeights, 10
predict.svmmaj, 10, 14
predict.transDat, 12
print.hinge (getHinge), 5
print.q.svmmaj, 12
print.summary.svmmaj (print.svmmaj), 15
print.svmmaj, 14, 15
print.svmmajcrossval, 16

roccurve, 17

summary.svmmaj (print.svmmaj), 15
summary.svmmajcrossval
  (print.svmmajcrossval), 16

supermarket1996, 17
svmmaj, 6–8, 10, 11, 13, 15, 16, 20, 23
svmmaj (print.q.svmmaj), 12
svmmajcrossval, 16, 19

transformdata, 21
unique, 13
voting, 21
X.svmmaj, 22