Package ‘evclass’

November 9, 2023

Type Package
Title Evidential Distance-Based Classification
Version 2.0.2
Date 2023-11-9
Author Thierry Denoeux
Maintainer Thierry Denoeux <tdenoeux@utc.fr>
Description Different evidential classifiers, which provide
outputs in the form of Dempster-Shafer mass functions. The methods are:
the evidential K-nearest neighbor rule, the evidential neural
network, radial basis function neural networks, logistic regression,
feed-forward neural networks.
License GPL-3
Depends R (>= 3.1.0)
Imports FNN, ibelief, R.utils
LazyData TRUE
Encoding UTF-8
RoxygenNote 7.2.3
VignetteBuilder knitr
Suggests knitr,rmarkdown,datasets,stats,nnet
NeedsCompilation no
Repository CRAN
Date/Publication 2023-11-09 11:00:02 UTC

R topics documented:

  calcAB ............................................................. 2
  calcm ............................................................ 3
  decision .......................................................... 5
  EkNNfit ......................................................... 7
  EkNNinit ....................................................... 8
**Description**

calcAB computes optimal coefficients alpha and beta needed to transform coefficients from logistic regression (or connections weights between the last hidden layer and the output layer of multilayer neural networks) into weights of evidence. These weights of evidence can then be used to express the outputs of logistic regression or multilayer neural networks as "latent" mass functions.

**Usage**

calcAB(W, mu = NULL)

**Arguments**

- **W**: Vector of coefficients of length \((d+1)\), where \(d\) is the number of features, in the case of \(M=2\) classes, or \((d+1,M)\) matrix of coefficients (or connection weights) in the case of \(M>2\) classes.

- **mu**: Optional vector containing the means of the \(d\) features.

**Value**

A list with two elements:

- **A**: Vector of length \(d\) (\(M=2\)) or matrix of size \((d,M)\) (for \(M>2\)) of coefficients alpha.

- **B**: Vector of length \(d\) (\(M=2\)) or matrix of size \((d,M)\) (for \(M>2\)) of coefficients beta.

**Author(s)**

Thierry Denoeux.
References


See Also

calcm

Examples

```r
## Example with 2 classes and logistic regression
data(ionosphere)
x <- ionosphere$x[, -2]
y <- ionosphere$y - 1
fit <- glm(y ~ x, family = 'binomial')
AB <- calcAB(fit$coefficients, colMeans(x))
AB

## Example with K>2 classes and multilayer neural network
library(nnet)
data(glass)
K <- max(glass$y)
d <- ncol(glass$x)
n <- nrow(x)
x <- scale(glass$x)
y <- as.factor(glass$y)
p <- 3 # number of hidden units
fit <- nnet(y ~ x, size = p) # training a neural network with 3 hidden units
W1 <- matrix(fit$wts[1:((p * (d + 1))], d + 1, p) # Input-to-hidden weights
W2 <- matrix(fit$wts[(p * (d + 1)) : ((p * (d + 1)) + K * (p + 1))], p + 1, K) # hidden-to-output weights
a1 <- cbind(rep(1, n), x) %*% W1 # hidden unit activations
o1 <- 1 / (1 + exp(-a1)) # hidden unit outputs
AB <- calcAB(W2, colMeans(o1))
AB
```

### Description

calcAB transforms coefficients alpha and beta computed by calcm into weights of evidence, and then into mass and contour (plausibility) functions. These mass functions can be used to express uncertainty about the prediction of logistic regression or multilayer neural network classifiers (See Denoeux, 2019).

### Usage

calcm(x, A, B)
Arguments

x Matrix (n,d) of feature values, where d is the number of features, and n is the number of observations. Can be a vector if d=1.

A Vector of length d (for M=2) or matrix of size (d,M) (for M>2) of coefficients alpha.

B Vector of length d (for M=2) or matrix of size (d,M) (for M>2) of coefficients beta.

Details

An error may occur if the absolute values of some coefficients are too high. It is then advised to recompute these coefficients by training the logistic regression or neural network classifier with L2 regularization. With M classes, the output mass functions have $2^M$ focal sets. Using this function with large M may cause memory issues.

Value

A list with six elements:

- F Matrix ($2^M$,M) of focal sets.
- mass Matrix (n,$2^M$) of mass functions (one in each row).
- pl Matrix (n,M) containing the plausibilities of singletons.
- bel Matrix (n,M) containing the degrees of belief of singletons.
- prob Matrix (n,M) containing the normalized plausibilities of singletons.
- conf Vector of length n containing the degrees of conflict.

Author(s)

Thierry Denoeux.

References


See Also

calcAB

Examples

```r
## Example with 2 classes and logistic regression
data(ionosphere)
x<-ionosphere$x[-2]
y<-ionosphere$y-1
fit<-glm(y ~ x,family="binomial")
AB<-calcAB(fit$coefficients,colMeans(x))
Bel<-calcm(x,AB$A,AB$B)
```
## Example with $K>2$ classes and multilayer neural network

```r
library(nnet)
data(glass)
K<-max(glass$y)
d<-ncol(glass$x)
n<-nrow(x)
x<-scale(glass$x)
y<-as.factor(glass$y)
p<-3 # number of hidden units
fit<-nnet(y~x,size=p) # training a neural network with 3 hidden units
W1<-matrix(fit$wts[1:(p*(d+1))],d+1,p) # Input-to-hidden weights
W2<-matrix(fit$wts[(p*(d+1)+1):(p*(d+1) + K*(p+1))],p+1,K) # hidden-to-output weights
a1<-cbind(rep(1,n),x)%*%W1 # hidden unit activations
o1<-1/(1+exp(-a1)) # hidden unit outputs
AB<-calcAB(W2,cbind(colMeans(o1)))
Bel<-calcm(o1,AB$A,AB$B)
Bel$focal
Bel$mass[1:5,]
Bel$pl[1:5,]
Bel$conf[1:5]
```

---

### decision

**Description**

decision returns decisions from a loss matrix and mass functions computed by an evidential classifier.

**Usage**

```r
decision(
  m,
  L = 1 - diag(ncol(m) - 1),
  rule = c("upper", "lower", "pignistic", "hurwicz"),
  rho = 0.5
)
```

**Arguments**

- **m**
  - Matrix of masses for n test cases. Each row is a mass function. The first M columns correspond to the mass assigned to each of the M classes. The last column corresponds to the mass assigned to the whole set of classes.

- **L**
  - The loss matrix of dimension (M,na) or (M+1,na), where na is the number of actions. $L[k,j]$ is the loss incurred if action j is chosen and the true class is $\omega_k$. If L has M+1 rows, the last row corresponds to the unknown class.
rule            Decision rule to be used. Must be one of these: 'upper' (upper expectation), 'lower' (lower expectations), 'pignistic' (pignistic expectation), 'hurwicz' (weighted sum of the lower and upper expectations).

rho             Parameter between 0 and 1. Used only if rule='hurwicz'.

Details

This function implements the decision rules described in Denoeux (1997), with an arbitrary loss function. The decision rules are the minimization of the lower, upper or pignistic expectation, and Jaffray’s decision rule based on minimizing a convex combination of the lower and upper expectations. The function also handles the case where there is an "unknown" class, in addition to the classes represented in the training set.

Value

A n-vector with the decisions (integers between 1 and na).

Author(s)

Thierry Denoeux.

References


See Also

EkNNval, proDSval

Examples

```r
## Example with M=2 classes
m<-matrix(c(0.9,0.1,0.4,0.6,0.1,0.8),3,3,byrow=TRUE)
## Loss matrix with na=4 acts: assignment to class 1, assignment to class 2, # rejection, and assignment to the unknown class.
L<-matrix(c(0.1,0.2,0.25,0.25,0.2,0.2,0.25,0.25,0.1,0.2,0.2,0.25,0.25,0),3,4)
d<-decision(m,L,'upper')  ## instances 2 and 3 are rejected
d<-decision(m,L,'lower')  ## instance 2 is rejected, instance 3 is # assigned to the unknown class
```
EkNNfit Training of the EkNN classifier

Description

EkNNfit optimizes the parameters of the EkNN classifier.

Usage

EkNNfit(
  x, 
  y,  
  K,  
  param = NULL,  
  alpha = 0.95,  
  lambda = 1/max(as.numeric(y)),  
  optimize = TRUE,  
  options = list(maxiter = 300, eta = 0.1, gain_min = 1e-06, disp = TRUE)
)

Arguments

x Input matrix of size n x d, where n is the number of objects and d the number of attributes.

y Vector of class labels (of length n). May be a factor, or a vector of integers from 1 to M (number of classes).

K Number of neighbors.

param Initial parameters (default: NULL).

alpha Parameter α (default: 0.95)

lambda Parameter of the cost function. If \( \lambda=1 \), the cost function measures the error between the plausibilities and the 0-1 target values. If \( \lambda=1/M \), where M is the number of classes (default), the pignistic probabilities are considered in the cost function. If \( \lambda=0 \), the beliefs are used.

optimize Boolean. If TRUE (default), the parameters are optimized.

options A list of parameters for the optimization algorithm: maxiter (maximum number of iterations), eta (initial step of gradient variation), gain_min (minimum gain in the optimisation loop), disp (Boolean; if TRUE, intermediate results are displayed during the optimization).

Details

If the argument param is not supplied, the function EkNNinit is called.
Value

A list with five elements:

- **param**: The optimized parameters.
- **cost**: Final value of the cost function.
- **err**: Leave-one-out error rate.
- **ypred**: Leave-one-out predicted class labels (coded as integers from 1 to M).
- **m**: Leave-one-out predicted mass functions. The first M columns correspond to the mass assigned to each class. The last column corresponds to the mass assigned to the whole set of classes.

Author(s)

Thierry Denoeux.

References


See Also

EkNNinit, EkNNval

Examples

```r
## Iris dataset
data(iris)
x <- iris[,1:4]
y <- iris[,5]
fit <-EkNNfit(x, y, K=5)
```

---

EkNNinit  
*Initialization of parameters for the EkNN classifier*

Description

EkNNinit returns initial parameter values for the EkNN classifier.

Usage

EkNNinit(x, y, alpha = 0.95)
EkNNinit

Arguments

x Input matrix of size n x d, where n is the number of objects and d the number of attributes.
y Vector of class labels (of length n). May be a factor, or a vector of integers from 1 to M (number of classes).
alpha Parameter \( \alpha \).

Details

Each parameter \( \gamma_k \) is set to the inverse of the square root of the mean Euclidean distances within class k. Note that \( \gamma_k \) here is the square root of the \( \gamma_k \) as defined in (Zouhal and Denoeux, 1998). By default, parameter alpha is set to 0.95. This value normally does not have to be changed.

Value

A list with two elements:

- **gamma** Vector of parameters \( \gamma_k \), of length c, the number of classes.
- **alpha** Parameter \( \alpha \), set to 0.95.

Author(s)

Thierry Denoeux.

References


See Also

EkNNfit, EkNNval

Examples

```r
## Iris dataset
data(iris)
x<-iris[,1:4]
y<-iris[,5]
param<-EkNNinit(x,y)
param
```
EkNNval

Classification of a test set by the EkNN classifier

Description

EkNNval classifies instances in a test set using the EkNN classifier.

Usage

EkNNval(xtrain, ytrain, xtst, K, ytst = NULL, param = NULL)

Arguments

- **xtrain**: Matrix of size ntrain x d, containing the values of the d attributes for the training data.
- **ytrain**: Vector of class labels for the training data (of length ntrain). May be a factor, or a vector of integers from 1 to M (number of classes).
- **xtst**: Matrix of size ntst x d, containing the values of the d attributes for the test data.
- **K**: Number of neighbors.
- **ytst**: Vector of class labels for the test data (optional). May be a factor, or a vector of integers from 1 to M (number of classes).
- **param**: Parameters, as returned by EkNNfit.

Details

If class labels for the test set are provided, the test error rate is also returned. If parameters are not supplied, they are given default values by EkNNinit.

Value

A list with three elements:

- **m**: Predicted mass functions for the test data. The first M columns correspond to the mass assigned to each class. The last column corresponds to the mass assigned to the whole set of classes.
- **ypred**: Predicted class labels for the test data (coded as integers from 1 to M).
- **err**: Test error rate.

Author(s)

Thierry Denoeux.

References


evclass

See Also
eKNNinit, EkNNfit

Examples

```
## Iris dataset
data(iris)
train<-sample(150,100)
xtrain<-iris[train,1:4]
ytrain<-iris[train,5]
x tst<-iris[-train,1:4]
ytst<-iris[-train,5]
K<-5
fit<-EkNNfit(xtrain,ytrain,K)
test<-EkNNval(xtrain,ytrain,xtst,K,ytst,fit$param)
```

evclass

evclass: A package for evidential classification

Description

The evclass package currently contains functions for three evidential classifiers: the evidential K-nearest neighbor (EK-NN) rule (Denoeux, 1995; Zouhal and Denoeux, 1998), the evidential neural network (Denoeux, 2000) and the RBF classifier with weight-of-evidence interpretation (Denoeux, 2019; Huang et al., 2022), as well as methods to compute output mass functions from trained logistic regression or multilayer classifiers as described in (Denoeux, 2019). In contrast with classical statistical classifiers, evidential classifiers quantify the uncertainty of the classification using Dempster-Shafer mass functions.

Details

The main functions are: EkNNinit, EkNNfit and EkNNval for the initialization, training and evaluation of the EK-NN classifier; proDSinit, proDSfit and proDSval for the evidential neural network classifier; decision for decision-making; RBFinit, RBFfit and RBFval for the RBF classifier; calcAB and calcm for computing output mass functions from trained logistic regression or multilayer classifiers.

References


See Also

EkNNinit, EkNNfit, EkNNval, proDSinit, proDSfit, proDSval, RBFinit, RBFfit and RBFval, decision, calcAB, calcm.

glass

Glass dataset

Description

This data set contains the description of 214 fragments of glass originally collected for a study in the context of criminal investigation. Each fragment has a measured reflectivity index and chemical composition (weight percent of Na, Mg, Al, Si, K, Ca, Ba and Fe). As suggested by Ripley (1994), 29 instances were discarded, and the remaining 185 were re-grouped in four classes: window float glass (70), window non-float glass (76), vehicle window glass (17) and other (22). The data set was split randomly in a training set of size 89 and a test set of size 96.

Usage

data(glass)

Format

A list with two elements:

- x The 185 x 9 object-attribute matrix.
- y A 185-vector containing the class labels.

References

P. M. Murphy and D. W. Aha. UCI Reposition of machine learning databases. [Machine readable data repository]. University of California, Departement of Information and Computer Science, Irvine, CA.


Examples

data(glass)
table(glass$y)
**ionosphere**

### Ionosphere dataset

**Description**

This dataset was collected by a radar system and consists of phased array of 16 high-frequency antennas with a total transmitted power of the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not. There are 351 instances and 34 numeric attributes. The first 175 instances are training data, the rest are test data. This version of dataset was used by Zouhal and Denoeux (1998).

**Usage**

```r
data(ionosphere)
```

**Format**

A list with two elements:

- `x` The 351 x 34 object-attribute matrix.
- `y` A 351-vector containing the class labels.

**References**

P. M. Murphy and D. W. Aha. UCI Repository of machine learning databases. [Machine readable data repository]. University of California, Department of Information and Computer Science, Irvine, CA.


**Examples**

```r
data(ionosphere)
table(vehicles$y)
```

---

**proDSfit**

*Training of the evidential neural network classifier*

**Description**

`proDSfit` performs parameter optimization for the evidential neural network classifier.
Usage

```r
proDSfit(
  x,
  y,
  param,
  lambda = 1/max(as.numeric(y)),
  mu = 0,
  optimProto = TRUE,
  options = list(maxiter = 500, eta = 0.1, gain_min = 1e-04, disp = 10)
)
```

**Arguments**

- `x`: Input matrix of size n x d, where n is the number of objects and d the number of attributes.
- `y`: Vector of class labels (of length n). May be a factor, or a vector of integers from 1 to M (number of classes).
- `param`: Initial parameters (see `link{proDSinit}`).
- `lambda`: Parameter of the cost function. If `lambda=1`, the cost function measures the error between the plausibilities and the 0-1 target values. If `lambda=1/M`, where M is the number of classes (default), the pignistic probabilities are considered in the cost function. If `lambda=0`, the beliefs are used.
- `mu`: Regularization hyperparameter (default=0).
- `optimProto`: Boolean. If TRUE, the prototypes are optimized (default). Otherwise, they are fixed.
- `options`: A list of parameters for the optimization algorithm: maxiter (maximum number of iterations), eta (initial step of gradient variation), gain_min (minimum gain in the optimisation loop), disp (integer; if >0, intermediate results are displayed every disp iterations).

**Details**

If `optimProto=TRUE` (default), the prototypes are optimized. Otherwise, they are fixed to their initial value.

**Value**

A list with three elements:

- `param`: Optimized network parameters.
- `cost`: Final value of the cost function.
- `err`: Training error rate.

**Author(s)**

 Thierry Denoeux.
Examples

```r
## Glass dataset
data(glass)
xapp<-glass$x[1:89,]
yapp<-glass$y[1:89]
x tst<-glass$x[90:185,]
ytst<-glass$y[90:185]
## Initialization
param0<-proDSinit(xapp,yapp,nproto=7)
## Training
fit<-proDSfit(xapp,yapp,param0)
```

### proDSinit

**Initialization of parameters for the evidential neural network classifier**

**Description**

proDSinit returns initial parameter values for the evidential neural network classifier.

**Usage**

```r
proDSinit(x, y, nproto, nprotoPerClass = FALSE, crisp = FALSE)
```

**Arguments**

- `x`: Input matrix of size n x d, where n is the number of objects and d the number of attributes.
- `y`: Vector of class labels (of length n). May be a factor, or a vector of integers from 1 to M (number of classes).
- `nproto`: Number of prototypes.
- `nprotoPerClass`: Boolean. If TRUE, there are nproto prototypes per class. If FALSE (default), the total number of prototypes is equal to nproto.
- `crisp`: Boolean. If TRUE, the prototypes have full membership to only one class. (Available only if nprotoPerClass=TRUE).
Details

The prototypes are initialized by the k-means algorithms. The initial membership values $u_{ik}$ of each prototype $p_i$ to class $\omega_k$ are normally defined as the proportion of training samples from class $\omega_k$ in the neighborhood of prototype $p_i$. If arguments crisp and nprotoPerClass are set to TRUE, the prototypes are assigned to one and only one class.

Value

A list with four elements containing the initialized network parameters

- **alpha** Vector of length r, where r is the number of prototypes.
- **gamma** Vector of length r
- **beta** Matrix of size (r,M), where M is the number of classes.
- **W** Matrix of size (r,d), containing the prototype coordinates.

Author(s)

Thierry Denoeux.

References


See Also

proDSfit, proDSval

Examples

```r
## Glass dataset
data(glass)
xapp<glass$x[1:89,]
yapp<glass$y[1:89]
param0<-proDSinit(xapp,yapp,nproto=7)
param0
```

```
proDSval(x, param, y = NULL)
```

Description

proDSval classifies instances in a test set using the evidential neural network classifier.

Usage

```r
proDSval(x, param, y = NULL)
```
Arguments

- **x**  
  Matrix of size n x d, containing the values of the d attributes for the test data.

- **param**  
  Neural network parameters, as provided by `proDSfit`.

- **y**  
  Optional vector of class labels for the test data. May be a factor, or a vector of integers from 1 to M (number of classes).

Details

If class labels for the test set are provided, the test error rate is also returned.

Value

A list with three elements:

- **m**  
  Predicted mass functions for the test data. The first M columns correspond to the mass assigned to each class. The last column corresponds to the mass assigned to the whole set of classes.

- **ypred**  
  Predicted class labels for the test data.

- **err**  
  Test error rate (if the class label of test data has been provided).

Author(s)

Thierry Denoeux.

References


See Also

`proDSinit`, `proDSfit`

Examples

```r
## Glass dataset
data(glass)
xapp<-'glass$x[1:89,]
yapp<-'glass$y[1:89]
xtst<-'glass$x[90:185,]
ytst<-'glass$y[90:185]
## Initialization
param0<-'proDSinit(xapp,yapp,nproto=7)
## Training
fit<-'proDSfit(xapp,yapp,param0)
## Test
val<-'proDSval(xtst,fit$param,ytst)
## Confusion matrix
table(ytst,val$ypred)
```
RBFfit

Training of a radial basis function classifier

Description

RBFfit performs parameter optimization for a radial basis function (RBF) classifier.

Usage

RBFfit(
  x,
  y,
  param,
  lambda = 0,
  control = list(fnscale = -1, trace = 2, maxit = 1000),
  optimProto = TRUE
)

Arguments

x
Input matrix of size n x d, where n is the number of objects and d the number of attributes.

y
Vector of class labels (of length n). May be a factor, or a vector of integers from 1 to M (number of classes).

param
Initial parameters (see RBFinit).

lambda
Regularization hyperparameter (default=0).

control
Parameters passed to function optim.

optimProto
Boolean. If TRUE, the prototypes are optimized (default). Otherwise, they are fixed.

Details

The RBF neural network is trained by maximizing the conditional log-likelihood (or, equivalently, by minimizing the cross-entropy loss function). The optimization procedure is the BFGS algorithm implemented in function optim.

Value

A list with three elements:

  param Optimized network parameters.
  loglik Final value of the log-likelihood objective function.
  err Training error rate.

Author(s)

Thierry Denoeux.
RBFinit

See Also

proDSinit, proDSval

Examples

## Glass dataset
data(glass)
xapp<-glass$x[1:89,]
yapp<-glass$y[1:89]
## Initialization
param0<-RBFinit(xapp,yapp,nproto=7)
## Training
fit<-RBFfit(xapp,yapp,param0,control=list(fnscale=-1,trace=2))

RBFinit

Initialization of parameters for a Radial Basis Function classifier

Description

RBFinit returns initial parameter values for a Radial Basis Function classifier.

Usage

RBFinit(x, y, nproto)

Arguments

x Input matrix of size n x d, where n is the number of objects and d the number of attributes.
y Vector of class labels (of length n). May be a factor, or a vector of integers from 1 to M (number of classes).
nproto Number of prototypes

Details

The prototypes are initialized by the k-means algorithms. The hidden-to-output weights are ini-
tialized by linear regression. The scale parameter for each prototype is computed as the inverse of
the square root of the mean squared distances to this prototype. The final number of prototypes
may be different from the desired number nproto depending on the result of the k-means clustering
(clusters composed of only one input vector are discarded).

Value

A list with three elements containing the initialized network parameters

P Matrix of size (R,d), containing the R prototype coordinates.
Gamma Vector of length R, containing the scale parameters.
W Matrix of size (R,M), containing the hidden-to-output weights.
Author(s)
Thierry Denoeux.

See Also
RBFfit, RBFval

Examples
## Glass dataset
data(glass)
xapp<-glass$x[1:89,]
yapp<-glass$y[1:89]
param0<-RBFinit(xapp,yapp,nproto=7)
param0

RBFval Classification of a test set by a radial basis function classifier

Description
RBFval classifies instances in a test set using a radial basis function classifier. Function calcm is called for computing output belief functions. It is recommended to set calc.belief=FALSE when the number of classes is very large, to avoid memory problems.

Usage
RBFval(x, param, y = NULL, calc.belief = TRUE)

Arguments
x Matrix of size n x d, containing the values of the d attributes for the test data.
param Neural network parameters, as provided by RBFfit.
y Optional vector of class labels for the test data. May be a factor, or a vector of integers from 1 to M (number of classes).
calc.belief If TRUE (default), output belief functions are calculated.

Details
If class labels for the test set are provided, the test error rate is also returned.

Value
A list with four elements:

ypred Predicted class labels for the test data.
err Test error rate (if the class label of test data has been provided).
Prob Output probabilities.
Belief If calc.belief=TRUE, output belief function, provided as a list output by function calcm.
**vehicles**

**Author(s)**
Thierry Denoeux.

**References**


**See Also**
- RBFinit, RBFfit, calcm

**Examples**

```r
## Glass dataset
data(glass)
xapp<-glass$x[1:89,]
yapp<-glass$y[1:89]
x tst<-glass$x[90:185,]
ytst<-glass$y[90:185]
## Initialization
param0<-RBFinit(xapp,yapp,nproto=7)
## Training
fit<-RBFfit(xapp,yapp,param0)
## Test
val<-RBFval(xtst,fit$param,ytst)
## Confusion matrix
table(ytst,val$ypred)
```

---

**Description**

This dataset was collected from silhouettes by the HIPS (Hierarchical Image Processing System) extension BINATTS Four model vehicles were used for the experiment: bus, Chevrolet van, Saab 9000 and Opel Manta. The data were used to distinguish 3D objects within a 2-D silhouette of the objects. There are 846 instances and 18 numeric attributes. The first 564 objects are training data, the rest are test data. This version of dataset was used by Zouhal and Denoeux (1998).

**Usage**

data(vehicles)
Format

A list with two elements:

x The 846 x 18 object-attribute matrix.

y A 846-vector containing the class labels.

References

P. M. Murphy and D. W. Aha. UCI Reposition of machine learning databases. [Machine readable data repository]. University of California, Departement of Information and Computer Science, Irvine, CA.


Examples

data(vehicles)
table(vehicles$y)
Index

* datasets
  glass, 12
  ionosphere, 13
  vehicles, 21

calcAB, 2, 4, 11, 12
calc, 3, 3, 11, 12, 20, 21

decision, 5, 11, 12
EkNNfit, 7, 9–12
EkNNinit, 7, 8, 8, 10–12
EkNNval, 6, 8, 9, 10, 11, 12
evclass, 11

glass, 12
ionosphere, 13

optim, 18

proDSfit, 11, 12, 13, 16, 17
proDSinit, 11, 12, 15, 15, 17, 19
proDSval, 6, 11, 12, 15, 16, 16, 19

RBFfit, 11, 12, 18, 20, 21
RBFinit, 11, 12, 18, 19, 21
RBFval, 11, 12, 20, 20

vehicles, 21