Package 'class'

April 26, 2020

Priority recommended

Version 7.3-17

Date 2020-04-26

Depends R (>= 3.0.0), stats, utils

Imports MASS

Description Various functions for classification, including k-nearest neighbour, Learning Vector Quantization and Self-Organizing Maps.

Title Functions for Classification

ByteCompile yes

License GPL-2 | GPL-3

URL http://www.stats.ox.ac.uk/pub/MASS4/

NeedsCompilation yes

Author Brian Ripley [aut, cre, cph], William Venables [cph]

Maintainer Brian Ripley <ripley@stats.ox.ac.uk>

Repository CRAN

Date/Publication 2020-04-26 10:31:55 UTC

R topics documented:

`batchSOM` *Self-Organizing Maps: Batch Algorithm*

Description

Kohonen's Self-Organizing Maps are a crude form of multidimensional scaling.

Usage

```
batchSOM(data, grid = somgrid(), radii, init)
```

Arguments

`data`	a matrix or data frame of observations, scaled so that Euclidean distance is appropriate.
`grid`	A grid for the representatives: see `somgrid`.
`radii`	the radii of the neighbourhood to be used for each pass: one pass is run for each element of `radii`.
`init`	the initial representatives. If missing, chosen (without replacement) randomly from `data`.

Details

The batch SOM algorithm of Kohonen(1995, section 3.14) is used.

Value

An object of class `"SOM"` with components

`grid`	the grid, an object of class `"somgrid"`.
`codes`	a matrix of representatives.

References

Kohonen, T. (1995) *Self-Organizing Maps.* Springer-Verlag.

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

`somgrid`, `SOM`

Examples

```
require(graphics)
data(crabs, package = "MASS")

lcrabs <- log(crabs[, 4:8])
crabs.grp <- factor(c("B", "b", "O", "o")[rep(1:4, rep(50,4))])
gr <- somgrid(topo = "hexagonal")
crabs.som <- batchSOM(lcrabs, gr, c(4, 4, 2, 2, 1, 1, 1, 0, 0))
plot(crabs.som)

bins <- as.numeric(knn1(crabs.som$code, lcrabs, 0:47))
plot(crabs.som$grid, type = "n")
symbols(crabs.som$grid$pts[, 1], crabs.som$grid$pts[, 2],
        circles = rep(0.4, 48), inches = FALSE, add = TRUE)
text(crabs.som$grid$pts[bins, ] + rnorm(400, 0, 0.1),
     as.character(crabs.grp))
```

---

condense *Condense training set for k-NN classifier*

Description

Condense training set for k-NN classifier

Usage

```
condense(train, class, store, trace = TRUE)
```

Arguments

`train`	matrix for training set
`class`	vector of classifications for test set
`store`	initial store set. Default one randomly chosen element of the set.
`trace`	logical. Trace iterations?

Details

The store set is used to 1-NN classify the rest, and misclassified patterns are added to the store set. The whole set is checked until no additions occur.

Value

Index vector of cases to be retained (the final store set).

References

P. A. Devijver and J. Kittler (1982) *Pattern Recognition. A Statistical Approach.* Prentice-Hall, pp. 119–121.

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

reduce.nn, multiedit

Examples

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
keep <- condense(train, cl)
knn(train[keep, , drop=FALSE], test, cl[keep])
keep2 <- reduce.nn(train, keep, cl)
knn(train[keep2, , drop=FALSE], test, cl[keep2])
```

---

knn *k-Nearest Neighbour Classification*

Description

k-nearest neighbour classification for test set from training set. For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the kth nearest vector, all candidates are included in the vote.

Usage

```
knn(train, test, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)
```

Arguments

train	matrix or data frame of training set cases.
test	matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case.
cl	factor of true classifications of training set
k	number of neighbours considered.
l	minimum vote for definite decision, otherwise doubt. (More precisely, less than k-l dissenting votes are allowed, even if k is increased by ties.)
prob	If this is true, the proportion of the votes for the winning class are returned as attribute prob.
use.all	controls handling of ties. If true, all distances equal to the kth largest are included. If false, a random selection of distances equal to the kth is chosen to use exactly k neighbours.

Value

Factor of classifications of test set. doubt will be returned as NA.

References

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

knn1, knn.cv

Examples

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
knn(train, test, cl, k = 3, prob=TRUE)
attributes(.Last.value)
```

---

knn.cv *k-Nearest Neighbour Cross-Validatory Classification*

Description

k-nearest neighbour cross-validatory classification from training set.

Usage

```
knn.cv(train, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)
```

Arguments

train	matrix or data frame of training set cases.
cl	factor of true classifications of training set
k	number of neighbours considered.
l	minimum vote for definite decision, otherwise doubt. (More precisely, less than k-l dissenting votes are allowed, even if k is increased by ties.)
prob	If this is true, the proportion of the votes for the winning class are returned as attribute prob.
use.all	controls handling of ties. If true, all distances equal to the kth largest are included. If false, a random selection of distances equal to the kth is chosen to use exactly k neighbours.

Details

This uses leave-one-out cross validation. For each row of the training set `train`, the `k` nearest (in Euclidean distance) other training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the kth nearest vector, all candidates are included in the vote.

Value

Factor of classifications of training set. `doubt` will be returned as `NA`.

References

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

knn

Examples

```
train <- rbind(iris3[,,1], iris3[,,2], iris3[,,3])
cl <- factor(c(rep("s",50), rep("c",50), rep("v",50)))
knn.cv(train, cl, k = 3, prob = TRUE)
attributes(.Last.value)
```

---

knn1 *1-Nearest Neighbour Classification*

Description

Nearest neighbour classification for test set from training set. For each row of the test set, the nearest (by Euclidean distance) training set vector is found, and its classification used. If there is more than one nearest, a majority vote is used with ties broken at random.

Usage

```
knn1(train, test, cl)
```

Arguments

`train`	matrix or data frame of training set cases.
`test`	matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case.
`cl`	factor of true classification of training set.

Value

Factor of classifications of test set.

References

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

knn

Examples

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
knn1(train, test, cl)
```

---

lvq1 *Learning Vector Quantization 1*

Description

Moves examples in a codebook to better represent the training set.

Usage

```
lvq1(x, cl, codebk, niter = 100 * nrow(codebk$x), alpha = 0.03)
```

Arguments

x	a matrix or data frame of examples
cl	a vector or factor of classifications for the examples
codebk	a codebook
niter	number of iterations
alpha	constant for training

Details

Selects niter examples at random with replacement, and adjusts the nearest example in the codebook for each.

Value

A codebook, represented as a list with components x and cl giving the examples and classes.

References

Kohonen, T. (1990) The self-organizing map. *Proc. IEEE* **78**, 1464–1480.

Kohonen, T. (1995) *Self-Organizing Maps.* Springer, Berlin.

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

`lvqinit`, `olvq1`, `lvq2`, `lvq3`, `lvqtest`

Examples

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
cd <- lvqinit(train, cl, 10)
lvqtest(cd, train)
cd0 <- olvq1(train, cl, cd)
lvqtest(cd0, train)
cd1 <- lvq1(train, cl, cd0)
lvqtest(cd1, train)
```

`lvq2` *Learning Vector Quantization 2.1*

Description

Moves examples in a codebook to better represent the training set.

Usage

```
lvq2(x, cl, codebk, niter = 100 * nrow(codebk$x), alpha = 0.03,
     win = 0.3)
```

Arguments

`x`	a matrix or data frame of examples
`cl`	a vector or factor of classifications for the examples
`codebk`	a codebook
`niter`	number of iterations
`alpha`	constant for training
`win`	a tolerance for the closeness of the two nearest vectors.

Details

Selects `niter` examples at random with replacement, and adjusts the nearest two examples in the codebook if one is correct and the other incorrect.

Value

A codebook, represented as a list with components `x` and `cl` giving the examples and classes.

References

Kohonen, T. (1990) The self-organizing map. *Proc. IEEE* **78**, 1464–1480.

Kohonen, T. (1995) *Self-Organizing Maps.* Springer, Berlin.

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

`lvqinit`, `lvq1`, `olvq1`, `lvq3`, `lvqtest`

Examples

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
cd <- lvqinit(train, cl, 10)
lvqtest(cd, train)
cd0 <- olvq1(train, cl, cd)
lvqtest(cd0, train)
cd2 <- lvq2(train, cl, cd0)
lvqtest(cd2, train)
```

---

lvq3 *Learning Vector Quantization 3*

Description

Moves examples in a codebook to better represent the training set.

Usage

```
lvq3(x, cl, codebk, niter = 100*nrow(codebk$x), alpha = 0.03,
     win = 0.3, epsilon = 0.1)
```

Arguments

`x`	a matrix or data frame of examples
`cl`	a vector or factor of classifications for the examples
`codebk`	a codebook
`niter`	number of iterations
`alpha`	constant for training
`win`	a tolerance for the closeness of the two nearest vectors.
`epsilon`	proportion of move for correct vectors

Details

Selects `niter` examples at random with replacement, and adjusts the nearest two examples in the codebook for each.

Value

A codebook, represented as a list with components `x` and `cl` giving the examples and classes.

References

Kohonen, T. (1990) The self-organizing map. *Proc. IEEE* **78**, 1464–1480.

Kohonen, T. (1995) *Self-Organizing Maps.* Springer, Berlin.

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

`lvqinit`, `lvq1`, `olvq1`, `lvq2`, `lvqtest`

Examples

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
cd <- lvqinit(train, cl, 10)
lvqtest(cd, train)
cd0 <- olvq1(train, cl, cd)
lvqtest(cd0, train)
cd3 <- lvq3(train, cl, cd0)
lvqtest(cd3, train)
```

`lvqinit` *Initialize a LVQ Codebook*

Description

Construct an initial codebook for LVQ methods.

Usage

```
lvqinit(x, cl, size, prior, k = 5)
```

Arguments

x	a matrix or data frame of training examples, n by p.
cl	the classifications for the training examples. A vector or factor of length n.
size	the size of the codebook. Defaults to `min(round(0.4*ng*(ng-1 + p/2),0),n)` where ng is the number of classes.
prior	Probabilities to represent classes in the codebook. Default proportions in the training set.
k	k used for k-NN test of correct classification. Default is 5.

Details

Selects size examples from the training set without replacement with proportions proportional to the prior or the original proportions.

Value

A codebook, represented as a list with components x and cl giving the examples and classes.

References

Kohonen, T. (1990) The self-organizing map. *Proc. IEEE* **78**, 1464–1480.

Kohonen, T. (1995) *Self-Organizing Maps.* Springer, Berlin.

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

lvq1, lvq2, lvq3, olvq1, lvqtest

Examples

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
cd <- lvqinit(train, cl, 10)
lvqtest(cd, train)
cd1 <- olvq1(train, cl, cd)
lvqtest(cd1, train)
```

`lvqtest` *Classify Test Set from LVQ Codebook*

Description

Classify a test set by 1-NN from a specified LVQ codebook.

Usage

```
lvqtest(codebk, test)
```

Arguments

`codebk`	codebook object returned by other LVQ software
`test`	matrix of test examples

Details

Uses 1-NN to classify each test example against the codebook.

Value

Factor of classification for each row of `x`

References

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

`lvqinit`, `olvq1`

Examples

```
# The function is currently defined as
function(codebk, test) knn1(codebk$x, test, codebk$cl)
```

multiedit *Multiedit for k-NN Classifier*

Description

Multiedit for k-NN classifier

Usage

```
multiedit(x, class, k = 1, V = 3, I = 5, trace = TRUE)
```

Arguments

x	matrix of training set.
class	vector of classification of training set.
k	number of neighbours used in k-NN.
V	divide training set into V parts.
I	number of null passes before quitting.
trace	logical for statistics at each pass.

Value

Index vector of cases to be retained.

References

P. A. Devijver and J. Kittler (1982) *Pattern Recognition. A Statistical Approach.* Prentice-Hall, p. 115.

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

condense, reduce.nn

Examples

```
tr <- sample(1:50, 25)
train <- rbind(iris3[tr,,1], iris3[tr,,2], iris3[tr,,3])
test <- rbind(iris3[-tr,,1], iris3[-tr,,2], iris3[-tr,,3])
cl <- factor(c(rep(1,25),rep(2,25), rep(3,25)), labels=c("s", "c", "v"))
table(cl, knn(train, test, cl, 3))
ind1 <- multiedit(train, cl, 3)
length(ind1)
table(cl, knn(train[ind1, , drop=FALSE], test, cl[ind1], 1))
ntrain <- train[ind1,]; ncl <- cl[ind1]
```

```
ind2 <- condense(ntrain, ncl)
length(ind2)
table(cl, knn(ntrain[ind2, , drop=FALSE], test, ncl[ind2], 1))
```

---

olvq1 *Optimized Learning Vector Quantization 1*

Description

Moves examples in a codebook to better represent the training set.

Usage

```
olvq1(x, cl, codebk, niter = 40 * nrow(codebk$x), alpha = 0.3)
```

Arguments

x	a matrix or data frame of examples
cl	a vector or factor of classifications for the examples
codebk	a codebook
niter	number of iterations
alpha	constant for training

Details

Selects niter examples at random with replacement, and adjusts the nearest example in the codebook for each.

Value

A codebook, represented as a list with components x and cl giving the examples and classes.

References

Kohonen, T. (1990) The self-organizing map. *Proc. IEEE* **78**, 1464–1480.

Kohonen, T. (1995) *Self-Organizing Maps.* Springer, Berlin.

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

lvqinit, lvqtest, lvq1, lvq2, lvq3

Examples

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
cd <- lvqinit(train, cl, 10)
lvqtest(cd, train)
cd1 <- olvq1(train, cl, cd)
lvqtest(cd1, train)
```

`reduce.nn` *Reduce Training Set for a k-NN Classifier*

Description

Reduce training set for a k-NN classifier. Used after `condense`.

Usage

```
reduce.nn(train, ind, class)
```

Arguments

`train`	matrix for training set
`ind`	Initial list of members of the training set (from `condense`).
`class`	vector of classifications for test set

Details

All the members of the training set are tried in random order. Any which when dropped do not cause any members of the training set to be wrongly classified are dropped.

Value

Index vector of cases to be retained.

References

Gates, G.W. (1972) The reduced nearest neighbor rule. *IEEE Trans. Information Theory* **IT-18**, 431–432.

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

`condense`, `multiedit`

Examples

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
keep <- condense(train, cl)
knn(train[keep,], test, cl[keep])
keep2 <- reduce.nn(train, keep, cl)
knn(train[keep2,], test, cl[keep2])
```

SOM *Self-Organizing Maps: Online Algorithm*

Description

Kohonen's Self-Organizing Maps are a crude form of multidimensional scaling.

Usage

```
SOM(data, grid = somgrid(), rlen = 10000, alpha, radii, init)
```

Arguments

data	a matrix or data frame of observations, scaled so that Euclidean distance is appropriate.
grid	A grid for the representatives: see somgrid.
rlen	the number of updates: used only in the defaults for alpha and radii.
alpha	the amount of change: one update is done for each element of alpha. Default is to decline linearly from 0.05 to 0 over rlen updates.
radii	the radii of the neighbourhood to be used for each update: must be the same length as alpha. Default is to decline linearly from 4 to 1 over rlen updates.
init	the initial representatives. If missing, chosen (without replacement) randomly from data.

Details

alpha and radii can also be lists, in which case each component is used in turn, allowing two- or more phase training.

Value

An object of class "SOM" with components

grid	the grid, an object of class "somgrid".
codes	a matrix of representatives.

References

Kohonen, T. (1995) *Self-Organizing Maps.* Springer-Verlag

Kohonen, T., Hynninen, J., Kangas, J. and Laaksonen, J. (1996) *SOM PAK: The self-organizing map program package.* Laboratory of Computer and Information Science, Helsinki University of Technology, Technical Report A31.

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

somgrid, batchSOM

Examples

```
require(graphics)
data(crabs, package = "MASS")

lcrabs <- log(crabs[, 4:8])
crabs.grp <- factor(c("B", "b", "O", "o")[rep(1:4, rep(50,4))])
gr <- somgrid(topo = "hexagonal")
crabs.som <- SOM(lcrabs, gr)
plot(crabs.som)

## 2-phase training
crabs.som2 <- SOM(lcrabs, gr,
    alpha = list(seq(0.05, 0, len = 1e4), seq(0.02, 0, len = 1e5)),
    radii = list(seq(8, 1, len = 1e4), seq(4, 1, len = 1e5)))
plot(crabs.som2)
```

---

somgrid *Plot SOM Fits*

Description

Plotting functions for SOM results.

Usage

```
somgrid(xdim = 8, ydim = 6, topo = c("rectangular", "hexagonal"))

## S3 method for class 'somgrid'
plot(x, type = "p", ...)

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

Arguments

xdim, ydim	dimensions of the grid
topo	the topology of the grid.
x	an object inheriting from class "somgrid" or "SOM".
type, ...	graphical parameters.

Details

The class "somgrid" records the coordinates of the grid to be used for (batch or on-line) SOM: this has a plot method.

The plot method for class "SOM" plots a stars plot of the representative at each grid point.

Value

For somgrid, an object of class "somgrid", a list with components

pts	a two-column matrix giving locations for the grid points.
xdim, ydim, topo	as in the arguments to somgrid.

References

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth edition. Springer.

See Also

batchSOM, SOM

Index