The kohonen Package
April 19, 2006

Version 1.0.1
Date 23/02/2006
Title Supervised and unsupervised self-organising maps
Author Ron Wehrens
Maintainer Ron Wehrens <r.wehrens@science.ru.nl>
Description Supervised and unsupervised self-organising maps
License GPL version 2 or newer
Depends class

R topics documented:

bdk ................................................................. 1
classvec2classmat ............................................. 3
nirTshift ........................................................ 4
plot.kohonen .................................................. 5
predict.kohonen .............................................. 7
som ............................................................ 8
somKmeans ..................................................... 10
unit.distances ................................................ 10
wines .......................................................... 11
xyf ............................................................. 11

Index 14

bdk Supervised version of Kohonen's self-organising maps

Description

Supervised version of self-organising maps for mapping high-dimensional spectra or patterns to 2D: the Bi-Directional Kohonen map. This is an alternating training of the X-space and the Y-space of the map, where for updating the X-space more weight is given to the features in Y, and vice versa. Weights start by default with values of (0.75, 0.25) and during training go to (0.5, 0.5). Prediction is done only using the X-space.
Usage

\[ \text{bdk}(\text{data}, \text{Y}, \text{grid} = \text{somgrid}(), \text{rlen} = 100, \text{alpha} = c(0.05, 0.01), \]
\[ \text{radius} = \text{quantile(nhbrdist, 0.67)}, \text{xweight} = 0.75, \]
\[ \text{toroidal} = \text{FALSE}, \text{keep.data} = \text{TRUE}) \]

Arguments

data a matrix, with each row representing an object.
Y property that is to be modelled. In case of classification, Y is a matrix of zeros, with exactly one '1' in each row indicating the class. For prediction of continuous properties, Y is a vector. A combination is possible, too, but one then should take care of appropriate scaling.
grid a grid for the representatives: see somgrid.
rlen the number of times the complete data set will be presented to the network.
alpha learning rate, a vector of two numbers indicating the amount of change. Default is to decline linearly from 0.05 to 0.01 over rlen updates.
radius the starting radius of the neighbourhood to be used for each update: the decrease is linear over rlen updates and reaches the value of 1 after one-third of the iterations. After that, only the winning units are updated.
xweight the initial weight given to the X map in the calculation of distances for updating Y, and to the Y map for updating X. This will linearly go to 0.5 during training. Defaults to 0.75.
toroidal if TRUE, the edges of the map are joined. Note that in a hexagonal toroidal map, the number of rows must be even.
keep.data save data in return value.

Value

an object of class "kohonen" with components

grid the grid, an object of class "somgrid".
changes matrix containing two columns of mean average deviations from code vectors. Column 1 contains deviations used for updating Y; column 2 for updating X.
codes matrix of code vectors.
codeYs matrix of Y values associated with each unit.
toroidal whether a toroidal map is used.
data data matrix, only if keep.data is TRUE.
Y Y, only if keep.data is TRUE.

Author(s)

Ron Wehrens

References


See Also

som, xyf, plot.kohonen, predict.kohonen
Examples

### Simulated example
library(MASS)
csize <- 15
c1 <- mvrnorm(csize, mu=c(5,3,4), Sigma=matrix(c(2, 0,0,0, 2, 1, 0, 1, 2),3,3))
c2 <- mvrnorm(csize, mu=c(5.5, 3.5, 4.5), Sigma=matrix(c(2, 0,0,0, 2, 1, 0, 1, 2),3,3))
c3 <- mvrnorm(csize, mu=c(0,0,0), Sigma=matrix(c(2, 0,0,0, 2, 1, 0, 1, 2),3,3))
X <- rbind(c1, c2, c3)
classes <- c(rep(1, csize), rep(2, csize), rep(3, csize))
bdkmap <- bdk(X, classvec2classmat(classes), somgrid(4, 4, "hexagonal"))
plot(bdkmap, "prediction")

### Wine example
data(wines)
set.seed(7)
training <- sample(length(wine.classes), 120)
Xtraining <- scale(wines[training,])
bdk.wines <- bdk(Xtraining, classvec2classmat(wine.classes[training]), grid = somgrid(5, 5, "hexagonal"), rlen=100)
Xtest <- scale(wines[-training,], center = attr(Xtraining, "scaled:center"), scale = attr(Xtraining, "scaled:scale"))
bdk.prediction <- predict(bdk.wines, data=Xtest)
bdkclass.prediction <- classmat2classvec(bdk.wines$codeYs[bdk.prediction$unit.classif,])
table(wine.classes[-training], bdkclass.prediction)

classvec2classmat Convert a classification vector into a matrix or the other way around.

Description

Functions toggle between a matrix representation, where class membership is indicated with one ‘1’ and for the rest zeros at each row, and an class vector (maybe integers or class names). The classification matrix contains one column per class. Conversion from a class matrix to a class vector assigns each row to the column with the highest value. An optional argument can be used to assign only those objects that have a probability higher than a certain threshold (default is 0).

Usage

classvec2classmat(yvec)
classmat2classvec(ymat, threshold=0)
nirTshift

Arguments

\texttt{yvec} \hspace{1cm} \text{class vector. Usually integer values, but other types are also allowed.}
\texttt{ymat} \hspace{1cm} \text{class matrix: every column corresponds to a class.}
\texttt{threshold} \hspace{1cm} \text{only classify into a class if the probability is larger than this threshold.}

Value

\texttt{classvec2classmat} returns the classification matrix, where each column consists of zeros and ones; \texttt{classmat2classvec} returns a class vector (integers).

Author(s)

Ron Wehrens

See Also

\texttt{bdk, xyf}

Examples

\begin{verbatim}
classes <- c(rep(1, 5), rep(2, 7), rep(3, 9))
classmat <- classvec2classmat(classes)
classmat
classmat2classvec(classmat)
\end{verbatim}

\section*{nirTshift}

\textit{Near-infrared data with temperature effects}

Description

Near-infrared spectra of ternary mixtures of ethanol, water and iso-propanol, measured at five different temperatures (30, 40, ..., 70 degrees Centigrade). A training set of 65 mixtures (13 spectra per temperature) is available, as well as a test set of 30 spectra (5 per temperature). Spectra are in variables \texttt{nirXtrain} and \texttt{nirXtest}; concentrations of the three compounds in \texttt{nirYtrain} and \texttt{nirYtest}.

References


Examples

\begin{verbatim}
data(nirTshift)
## First model concentrations, all at once
set.seed(3)
nirnet <- xyf(data = nirXtr, Y = nirYtr, xweight=.75, 
grid = somgrid(6, 6, "hexagonal"), rlen=500)
par(mfrow = c(1,3))
plot(nirnet, "predict")
## Concentrate on water, compound 2:
par(mfrow = c(1,2))
\end{verbatim}
set.seed(3)
nirnet <- xyf(data = nirXtr, Y = nirYtr[,2], xweight=.75, grid = somgrid(6, 6, "hexagonal"), rlen=500)
plot(nirnet, "predict", main="Prediction of water content")
## Plot temperatures as circles
temps <- rep(seq(30, 70, by=10), times=rep(13,5))
symbols(nirnet$grid$pts[nirnet$classif,] +
       matrix(rnorm(nrow(nirYtr)*2, sd=.1), ncol=2),
circles = (temps - 20)/250,
inches = FALSE, add = TRUE)
## Model temperatures
set.seed(3)
nirnet2 <- xyf(data = nirXtr, Y = classvec2classmat(temps), xweight=.3, grid = somgrid(6, 6, "hexagonal"), rlen=500)
plot(nirnet2, "predict", palette.name = rainbow, main="Prediction of temperatures")
## Plot concentrations of water as circles
symbols(nirnet2$grid$pts[nirnet2$classif,] +
       matrix(rnorm(nrow(nirYtr)*2, sd=.1), ncol=2),
circles = 0.05 + 0.4 * nirYtr[,2],
inches = FALSE, add = TRUE)

---

**plot.kohonen**  
Plot kohonen object

**Description**  
Plot self-organising map, obtained from function kohonen. Several types of plots are supported.

**Usage**  
plot.kohonen(x, type = c("changes", "codes", "counts", "mapping", "prediction", "property"), classif, labels=NULL, pchs=NULL, main=NULL, palette.name = heat.colors, ncolors, zlim=NULL, property, heatkey=TRUE, contin, ...)

**Arguments**  
x  
  kohonen object.

type  
  type of plot. (Wow!)

classif  
  classification object, as returned by predict.kohonen, or vector of unit numbers. Only needed if type equals "classifications" and "counts".

labels  
  labels to plot when type equals "classifications".

pchs  
  symbols to plot when type equals "classifications".

main  
  title of the plot.

palette.name  
  colors to use as unit background for "codes", "counts", "prediction", and "property" plotting types.

ncolors  
  number of colors to use. Default is 20 for continuous data, and the number of distinct values (if less than 20) for class data.
zlim optional range for color coding of unit backgrounds.
property values to use with the "property" plotting type.
heatkey whether or not to generate a heatkey at the left side of the plot in the "property" and "counts" plotting types.
contin whether or not the data should be seen as discrete (i.e. classes) or continuous in nature. Only relevant for the colour keys of plots of supervised networks.
... other graphical parameters, e.g. colours of labels, or plotting symbols, in the "classifications" plotting type.

Details

Several different types of plots are supported:

"changes" shows the changes in similarity (on a more or less arbitrary scale) during training.
"codes" shows the unit vectors.
"counts" shows the number of objects mapped to the individual units. Empty units are depicted in gray.
"mapping" show where objects are mapped. It needs the "classif" argument, and a "labels" or "pchs" argument.
"prediction" for supervised maps, the class or predicted value for each unit.
"property" properties of each unit can be calculated and shown in colour code. It can be used to visualise the similarity of one particular object to all units in the map, to show the mean similarity of all units and the objects mapped to them, etcetera. The parameter property contains the numerical values. See examples below.

Author(s)

Ron Wehrens

See Also

som, bdk.xyf

Examples

data(wines)
set.seed(7)

kohmap <- xyf(scale(wines), classvec2classmat(wine.classes),
               grid = somgrid(5, 5, "hexagonal"), rlen=100)
plot(kohmap, type="changes", main="plot of changes")
plot(kohmap, type="codes", main="codes plot")
plot(kohmap, type="counts", main="counts plot")
plot(kohmap, type="mapping",
     labels=wine.classes, col=wine.classes,
     main="mapping plot")
plot(kohmap, type="mapping",
     pchs=wine.classes, col=wine.classes,
     main="another mapping plot")
### Made-up labels :-) 
plot(kohmap, type="prediction",
predict.kohonen

    labels=c("Chianti", "Barolo", "Montepulciano"),
    palette.name = rainbow,
    main="unit class prediction", cex=.8)

### Plot mean similarity of mapped objects to their unit codebook vector
mapping <- predict(kohmap, classif=kohmap$classif)
hits <- sort(unique(kohmap$classif))
similarities <- rep(NA, 25)
for (i in seq(along=hits))
    similarities[hits[i]] <- mean(mapping$distances[kohmap$classif == hits[i]])

plot(kohmap, type="property", property=similarities, main="property plot")

---

**predict.kohonen**  
*Classify using a trained Kohonen map*

**Description**

Map objects to a trained Kohonen map. The map may be obtained by supervised methods (`xyf`, `bdk`), but only the X-space is considered in prediction. If the classification is already known, the function returns the similarity for each object to the unit it is mapped to.

**Usage**

`predict.kohonen(object, ...)`

**Arguments**

- **object**
  
  trained network.

- **...**

  optional arguments. Two arguments in particular are important: if a data argument is given (a data matrix with the same number of variables as the codebook vectors of the network), predictions for these objects are returned, both the units at which the objects are mapped and the distance to these units. If a classification vector is given (argument `classif`), the distances to the corresponding codebook vectors are calculated.

**Value**

Returns a list with components

- **unit.classif**

  unit numbers to which objects in the data matrix are mapped.

- **classif**

  classes associated with the units (only for the supervised `bdk` and `xyf` networks).

- **distances**

  similarities, expressed as distances, of objects to codebook vectors.

**Author(s)**

Ron Wehrens

**See Also**

`som, xyf, bdk`
Examples

```r
### Simulated example
library(MASS)

csize <- 15
c1 <- mvrnorm(csize, mu=c(5,3,4),
              Sigma=matrix(c(2, 0, 0, 2, 1, 0, 1, 2, 3, 3),3,3))
c2 <- mvrnorm(csize, mu=c(5.5, 3.5, 4.5),
              Sigma=matrix(c(2, 0, 0, 2, 1, 0, 1, 2, 3, 3),3,3))
c3 <- mvrnorm(csize, mu=c(0,0,0),
              Sigma=matrix(c(2, 0, 0, 2, 1, 0, 1, 2, 3, 3),3,3))

X <- rbind(c1, c2, c3)
classes <- c(rep(1, csize), rep(2, csize), rep(3, csize))

bdkmod <- bdk(X, classvec2classmat(classes), somgrid(4, 4, "hexagonal"))
predict(bdkmod)
```

som  

**Kohonen’s self-organising maps**

Description

Self-organising maps for mapping high-dimensional spectra or patterns to 2D; the Euclidean distance is used. Modelled after the SOM function in package class.

Usage

```r
som(data, grid=somgrid(), rlen = 100, alpha = c(0.05, 0.01),
    radius = quantile(nhbrdist, 0.67), init,
    toroidal = FALSE, FineTune = TRUE, keep.data = TRUE)
```

Arguments

- `data`: a matrix, with each row representing an object.
- `grid`: a grid for the representatives: see `somgrid`.
- `rlen`: the number of times the complete data set will be presented to the network.
- `alpha`: learning rate, a vector of two numbers indicating the amount of change. Default is to decline linearly from 0.05 to 0.01 over `rlen` updates.
- `radius`: the initial radius of the neighbourhood to be used for each update: it decreases linearly to 1 over `rlen` updates. The default is to start with a value that covers 2/3 of all units.
- `init`: the initial representatives, represented as a matrix. If missing, chosen (without replacement) randomly from `data`.
- `toroidal`: if TRUE, the edges of the map are joined. Note that in a hexagonal toroidal map, the number of rows must be even.
- `FineTune`: apply kmeans for fine-tuning the codebook vectors.
- `keep.data`: save data in return value.
Value

an object of class "kohonen" with components

grid the grid, an object of class "somgrid".
changes vector of mean average deviations from code vectors.
codes a matrix of code vectors.
classif winning units for all data objects.
toroidal whether a toroidal map is used.
data data matrix, only if keep.data is TRUE.

Author(s)

Ron Wehrens

References


See Also

xyf, bdk, plot.kohonen

Examples

data(wines)
set.seed(7)

training <- sample(length(wine.classes), 120)
Xtraining <- scale(wines[training,])
som.wines <- som(Xtraining, grid = somgrid(5, 5, "hexagonal"), rlen=100)

Xtest <- scale(wines[-training,],
    center = attr(Xtraining, "scaled:center"),
    scale = attr(Xtraining, "scaled:scale"))
som.classes <- rep(0, 25)
for (i in 1:25) {
    mappedHere <- which(som.wines$classif == i)
    if (length(mappedHere) > 0) {
        classes.mappedHere <- wine.classes[training[mappedHere]]
        som.classes[i] <-
            names(table(classes.mappedHere))[table(classes.mappedHere) ==
                max(table(classes.mappedHere))][1]
    }
}
som.prediction <- predict(som.wines, data=Xtest)
somclass.prediction <- som.classes[som.prediction$unit.classif]
table(wine.classes[-training], somclass.prediction)
somKmeans

**Refinement of codebook vectors by kmeans**

**Description**

Do kmeans of the codebook vectors, which is equivalent to the SOM algorithm when the size of the neighbourhood is smaller than 1. Not meant to be called directly.

**Usage**

```r
somKmeans(codes, data, max.iter = 20, verbose = FALSE)
```

**Arguments**

- `codes` codebook matrix for the X space: one row per unit.
- `data` training data, usually.
- `max.iter` default is 20, but convergence usually is much faster. If more iterations are needed, a warning is printed.
- `verbose` print more info to the screen.

**Value**

Returns a list with components

- `codes` new codebook matrix.
- `classif` classification vector.
- `niter` number of iterations needed for kmeans.

**Author(s)**

Ron Wehrens

---

unit.distances

**Calculate distances between units in a SOM**

**Description**

Calculate distances between units in a SOM.

**Usage**

```r
unit.distances(grid, toroidal)
```

**Arguments**

- `grid` an object of class `somgrid`.
- `toroidal` if true, edges of the map are joined so that the topology is that of a torus.
Value

Returns a matrix containing distances.

Author(s)

Ron Wehrens

Examples

```r
library(kohonen)
data(wines)
kohmap <- som(wines, grid = somgrid(5, 5, "hexagonal"), rlen=100)
par(mfrow=c(1,2))
dists <- unit.distances(kohmap$grid, toroidal=FALSE)
plot(kohmap, type="property", property=dists[1,],
     main="Distances to unit 1", zlim=c(0,4))
dists <- unit.distances(kohmap$grid, toroidal=TRUE)
plot(kohmap, type="property", property=dists[1,],
     main="Distances to unit 1 (toroidal)", zlim=c(0,4))
```

Description

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

Usage

data(wines)

Source

http://kdd.ics.uci.edu

Description

Supervised version of self-organising maps for mapping high-dimensional spectra or patterns to 2D. The name stands for X-Y fused SOMs. One vector for each object is created by concatenating X and Y, and a SOM is trained in the usual way, with one exception: the distance of an object to a unit is the sum of separate distances for X and Y spaces. Prediction is done only using the X-space.
Usage

\texttt{xyf(data, Y, grid=somgrid(), rlen = 100, alpha = c(0.05, 0.01),
radius = quantile(nhbrdist, 0.67),
xweight = 0.5, toroidal = FALSE, keep.data = TRUE)}

Arguments

data a matrix, with each row representing an object.
Y property that is to be modelled. In case of classification, \(Y\) is a matrix of zeros, with exactly one ‘1’ in each row indicating the class. For prediction of continuous properties, \(Y\) is a vector. A combination is possible, too, but one then should take care of appropriate scaling.
grid a grid for the representatives: see \texttt{somgrid}.
rlen the number of times the complete data set will be presented to the network.
alpha learning rate, a vector of two numbers indicating the amount of change. Default is to decline linearly from 0.05 to 0.01 over \(rlen\) updates.
radius the starting radius of the neighbourhood to be used for each update: the decrease is linear over \(rlen\) updates and reaches the value of 1 after one-third of the iterations.
xweight the weight given to the X map in the calculation of distances for updating \(Y\). Default is 0.5.
toroidal if TRUE, the edges of the map are joined. Note that in a hexagonal toroidal map, the number of rows must be even.
keep.data save data in return value.

Value

an object of class "kohonen" with components

\texttt{grid} the grid, an object of class "somgrid".
\texttt{changes} vector of mean average deviations from code vectors
\texttt{codes} a matrix of code vectors.
\texttt{codeYs} a matrix of \(Y\) values associated with each unit.
\texttt{toroidal} whether a toroidal map is used.
\texttt{data} data matrix, only if \texttt{keep.data} is TRUE.
\texttt{Y} \(Y\), only if \texttt{keep.data} is TRUE.

Author(s)

Ron Wehrens

References


See Also

\texttt{som, bdk, plot.kohonen, predict.kohonen}
### Simulated example

```r
library(MASS)

csize <- 150
c1 <- mvrnorm(csize, mu=c(5,3,4),
              Sigma=matrix(c(2, 0,0,0, 2, 1, 0, 1, 2),3,3))
c2 <- mvrnorm(csize, mu=c(5.5, 3.5, 4.5),
              Sigma=matrix(c(2, 0,0,0, 2, 1, 0, 1, 2),3,3))
c3 <- mvrnorm(csize, mu=c(0,0,0),
              Sigma=matrix(c(2, 0,0,0, 2, 1, 0, 1, 2),3,3))

X <- rbind(c1, c2, c3)
classes <- c(rep(1, csize), rep(2, csize), rep(3, csize))

xyfmap <- xyf(X, classvec2classmat(classes), somgrid(8, 8, "hexagonal"))
plot(xyfmap, "prediction")
```

### Wine example

```r
data(wines)
set.seed(7)

training <- sample(length(wine.classes), 120)
Xtraining <- scale(wines[training,])

xyf.wines <- xyf(Xtraining, classvec2classmat(wine.classes[training]),
                  grid = somgrid(5, 5, "hexagonal"), rlen=100)

Xtest <- scale(wines[-training,],
               center = attr(Xtraining, "scaled:center"),
               scale = attr(Xtraining, "scaled:scale"))

xyf.prediction <- predict(xyf.wines, data=Xtest)

xyfclass.prediction <- classmat2classvec(xyf.wines$codeYs[xyf.prediction$unit.classif,])

table(wine.classes[-training], xyfclass.prediction)
```
Index

*Topic classif*
  bdk, 1
  classvec2classmat, 3
  plot.kohonen, 5
  predict.kohonen, 7
  som, 8
  somKmeans, 10
  unit.distances, 10
  xyf, 11
*Topic datasets*
  nirTshift, 4
  wines, 11

bdk, 1, 4, 6, 7, 9, 12

classmat2classvec
  (classvec2classmat), 3
  classvec2classmat, 3

nirTshift, 4
nirXtest (nirTshift), 4
nirXtr (nirTshift), 4
nirYtest (nirTshift), 4
nirYtr (nirTshift), 4

plot.heatkey (plot.kohonen), 5
plot.kohchanges (plot.kohonen), 5
plot.kohcodes (plot.kohonen), 5
plot.kohcounts (plot.kohonen), 5
plot.kohmapping (plot.kohonen), 5
plot.kohonen, 2, 5, 9, 12
plot.kohpred (plot.kohonen), 5
plot.kohprop (plot.kohonen), 5
plot.somgrid (plot.kohonen), 5
predict.kohonen, 2, 7, 12

som, 2, 6, 7, 8, 12
somKmeans, 10

unit.distances, 10

wine.classes (wines), 11
wines, 11

xyf, 2, 4, 6, 7, 9, 11

14