Package ‘kknn’

October 13, 2022

Title Weighted k-Nearest Neighbors
Version 1.3.1
Date 2016-03-26
Description Weighted k-Nearest Neighbors for Classification, Regression and Clustering.
Depends R (>= 2.10)
Imports igraph (>= 1.0), Matrix, stats, graphics
ByteCompile TRUE
License GPL (>= 2)
NeedsCompilation yes
URL https://github.com/KlausVigo/kknn
Repository CRAN
Author Klaus Schliep [aut, cre],
    Klaus Hechenbichler [aut],
    Antoine Lizee [ctb]
Maintainer Klaus Schliep <klaus.schliep@gmail.com>
Date/Publication 2016-03-26 22:02:21

R topics documented:

  kknn-package ......................................................... 2
  contr.dummy ......................................................... 2
  glass ............................................................... 3
  ionosphere .......................................................... 4
  knn ................................................................. 5
  kknn-deprecated .................................................... 7
  miete ............................................................... 8
  specClust .......................................................... 10
  train.kknn .......................................................... 12

Index 15
Introduction

Weighted k-Nearest Neighbors Classification, Regression and spectral Clustering

The complete list of functions can be displayed with `library(help = kknn)`.

Author(s)

Klaus Schliep
Klaus Hechenbichler
Maintainer: Klaus Schliep <klaus.schliep@gmail.com>

References


contr.dummy

Contrast Matrices

Description

Returns a matrix of contrasts.

Usage

```r
contr.dummy(n, contrasts = TRUE)
contr.ordinal(n, contrasts = TRUE)
contr.metric(n, contrasts = TRUE)
```

Arguments

- `n` A vector containing levels of a factor, or the number of levels.
- `contrasts` A logical value indicating whether contrasts should be computed.

Details

`contr.dummy` is standard dummy-coding, `contr.metric` has the same effect like `as.numeric` (makes sense of course only for ordered variables). `contr.ordinal` computes contrasts for ordinal variables.
glass

Value
A matrix with \( n \) rows and \( n-1 \) columns for \texttt{contr.ordinal}, a matrix with \( n \) rows and \( n \) columns for \texttt{contr.dummy} and a vector of length \( n \) for \texttt{contr.metric}.

Author(s)
Klaus P. Schliep <klaus.schliep@gmail.com>

References

See Also
\texttt{contrasts}, \texttt{contr.poly} and \texttt{contr.sdif}

Examples
\begin{verbatim}
contr.metric(5)
contr.ordinal(5)
contr.dummy(5)
\end{verbatim}

A data frame with 214 observations, where the problem is to predict the type of glass in terms of their oxide content (i.e. Na, Fe, K, etc). The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence... if it is correctly identified!

Usage
data(glass)

Format
A data frame with 214 observations on the following 11 variables.

\begin{verbatim}
Id  Id number.
RI  Refractive index.
Na  Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10).
Mg  Magnesium.
Al  Aluminum.
\end{verbatim}
Si  Silicon.
K  Potassium.
Ca  Calcium.
Ba  Barium.
Fe  Iron.

Type  Type of glass: (class attribute)
   1 building windows float processed
   2 building windows non float processed
   3 vehicle windows float processed
   4 vehicle windows non float processed (none in this database)
   5 containers
   6 tableware
   7 headlamps

Source
• Creator: B. German, Central Research Establishment, Home Office Forensic Science Service, Aldermaston, Reading, Berkshire RG7 4PN
• Donor: Vina Spiehler, Ph.D., DABFT, Diagnostic Products Corporation

The data have been taken from the UCI Machine Learning Database Repository
http://www.ics.uci.edu/~mlearn/MLRepository.html
and were converted to R format by <klaus.schliep@gmail.com>.

Examples
   data(glass)
   str(glass)

---

ionosphere  Johns Hopkins University Ionosphere Database

Description
This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased
array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts.
See the paper for more details. 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; their signals pass through the ionosphere.

Received signals were processed using an autocorrelation function whose arguments are the time of
a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in
this database are described by 2 attributes per pulse number, corresponding to the complex values
returned by the function resulting from the complex electromagnetic signal.

Usage
   data(ionosphere)
**Format**

A data frame with 351 observations on the following 35 variables. The first 34 continuous covari-
ables are used for the prediction. The 35th attribute is either g ("good") or b ("bad") according to
the definition summarized above. This is a binary classification task.

**Source**

Vince Sigillito (vgs@aplce.pl.jhu.edu), Space Physics Group, Applied Physics Laboratory, Johns
Hopkins University, Johns Hopkins Road, Laurel, MD 20723

The data have been taken from the UCI Machine Learning Database Repository
http://www.ics.uci.edu/~mllearn/MLRepository.html
and were converted to R format by <klaus.schliep@gmail.com >.

**Examples**

data(ionosphere)

---

**kkn**

*Weighted k-Nearest Neighbor Classifier*

**Description**

Performs k-nearest neighbor classification of a test set using a training set. For each row of the
test set, the k nearest training set vectors (according to Minkowski distance) are found, and the
classification is done via the maximum of summed kernel densities. In addition even ordinal and
continuous variables can be predicted.

**Usage**

```r
kkn(formula = formula(train), train, test, na.action = na.omit(),
k = 7, distance = 2, kernel = "optimal", ykernel = NULL, scale=TRUE,
contrasts = c('unordered' = "contr.dummy", ordered = "contr.ordinal"))
kkn.dist(learn, valid, k = 10, distance = 2)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formula</td>
<td>A formula object.</td>
</tr>
<tr>
<td>train</td>
<td>Matrix or data frame of training set cases.</td>
</tr>
<tr>
<td>test</td>
<td>Matrix or data frame of test set cases.</td>
</tr>
<tr>
<td>learn</td>
<td>Matrix or data frame of training set cases.</td>
</tr>
<tr>
<td>valid</td>
<td>Matrix or data frame of test set cases.</td>
</tr>
<tr>
<td>na.action</td>
<td>A function which indicates what should happen when the data contain 'NA's.</td>
</tr>
<tr>
<td>k</td>
<td>Number of neighbors considered.</td>
</tr>
<tr>
<td>distance</td>
<td>Parameter of Minkowski distance.</td>
</tr>
</tbody>
</table>
kernel: Kernel to use. Possible choices are "rectangular" (which is standard unweighted knn), "triangular", "epanechnikov" (or beta(2,2)), "biweight" (or beta(3,3)), "triweight" (or beta(4,4)), "cos", "inv", "gaussian", "rank" and "optimal".

y_kernel: Window width of an y-kernel, especially for prediction of ordinal classes.

scale: logical, scale variable to have equal sd.

contrasts: A vector containing the 'unordered' and 'ordered' contrasts to use.

Details

This nearest neighbor method expands knn in several directions. First it can be used not only for classification, but also for regression and ordinal classification. Second it uses kernel functions to weight the neighbors according to their distances. In fact, not only kernel functions but every monotonic decreasing function $f(x)\forall x > 0$ will work fine.

The number of neighbours used for the "optimal" kernel should be $\left\lceil \frac{2(d + 4)}{(d + 2)}\frac{d}{(d + 4)}k \right\rceil$, where k is the number that would be used for unweighted knn classification, i.e. kernel="rectangular". This factor $(2(d + 4)/(d + 2))d/(d + 4))$ is between 1.2 and 2 (see Samworth (2012) for more details).

Value

kknn returns a list-object of class kknn including the components

- fitted.values: Vector of predictions.
- CL: Matrix of classes of the k nearest neighbors.
- W: Matrix of weights of the k nearest neighbors.
- D: Matrix of distances of the k nearest neighbors.
- C: Matrix of indices of the k nearest neighbors.
- prob: Matrix of predicted class probabilities.
- response: Type of response variable, one of continuous, nominal or ordinal.
- distance: Parameter of Minkowski distance.
- call: The matched call.
- terms: The 'terms' object used.

Author(s)

Klaus P. Schliep <klaus.schliep@gmail.com>
Klaus Hechenbichler

References


See Also

train.kknn, simulation, knn and knn1

Examples

library(kknn)

data(iris)
m <- dim(iris)[1]
val <- sample(1:m, size = round(m/3), replace = FALSE, 
prob = rep(1/m, m))
iris.learn <- iris[-val,]
iris.valid <- iris[val,]
iris.kknn <- kknn(Species~., iris.learn, iris.valid, distance = 1, 
kernel = "triangular")
summary(iris.kknn)
fit <- fitted(iris.kknn)
table(iris.valid$Species, fit)

pcol <- as.character(as.numeric(iris.valid$Species))
pairs(iris.valid[1:4], pch = pcol, col = c("green3", "red")
[(iris.valid$Species != fit)+1])

data(ionosphere)
ionosphere.learn <- ionosphere[1:200,]
ionosphere.valid <- ionosphere[-c(1:200),]
fit.kknn <- kknn(class ~ ., ionosphere.learn, ionosphere.valid)
table(ionosphere.valid$class, fit.kknn$fit)

(fit.train1 <- train.kknn(class ~ ., ionosphere.learn, kmax = 15, 
kernel = c("triangular", "rectangular", "epanechnikov", "optimal"), distance = 1))
table(predict(fit.train1, ionosphere.valid), ionosphere.valid$class)

(fit.train2 <- train.kknn(class ~ ., ionosphere.learn, kmax = 15, 
kernel = c("triangular", "rectangular", "epanechnikov", "optimal"), distance = 2))
table(predict(fit.train2, ionosphere.valid), ionosphere.valid$class)

### kknn-deprecated

**Deprecated Functions in Package kknn**

**Description**

These functions are provided for compatibility with older versions of R only, and may be defunct as soon as of the next release.

**Usage**

simulation(formula, data, runs = 10, train = TRUE, k = 11, ...)
Arguments

- **formula**: A formula object.
- **data**: Matrix or data frame.
- **runs**: Number of crossvalidation runs.
- **train**: A logical value. If TRUE the training procedure for selecting optimal values of \(k\) and kernel is performed.
- **k**: Number or maximal number of neighbors considered, dependent of choice for train.
- **...**: Further arguments passed to or from other methods.

Value

A matrix, containing the mean and variance of the misclassification error, the absolute and the squared distances.

References


See Also

- Defunct and Deprecated

Description

Many german cities compose so-called rent standards to make a decision making instrument available to tenants, landlords, renting advisory boards and experts. The rent standards are used in particular for the determination of the local comparative rent (i.e. net rent as a function of household size, equipment, year of construction, etc.). For the composition of the rent standards, a representative random sample is drawn from all relevant households, and the interesting data are determined by interviewers by means of questionnaires. The dataset contains the data of 1082 households interviewed for the munich rent standard 1994.

Usage

```r
data(miete)
```
Format

A data frame with 1082 observations on the following 18 variables.

\textbf{nm} Net rent in DM.
\textbf{wfl} Floor space in sqm.
\textbf{bj} Year of construction.
\textbf{bad0} Bathroom in apartment?
\hspace{1em} 1 : no
\hspace{1em} 0 : yes
\textbf{zh} Central heating?
\hspace{1em} 1 : yes
\hspace{1em} 0 : no
\textbf{ww0} Hot water supply?
\hspace{1em} 1 : no
\hspace{1em} 0 : yes
\textbf{badkach} Tiled bathroom?
\hspace{1em} 1 : yes
\hspace{1em} 0 : no
\textbf{fenster} Window type:
\hspace{1em} 1 : plain windows
\hspace{1em} 0 : state-of-the-art windows
\textbf{kueche} Kitchen type
\hspace{1em} 1 : well equipped kitchen
\hspace{1em} 0 : plain kitchen
\textbf{mvdauer} Lease duration in years.
\textbf{bjkat} Age category of the building (bj categorized)
\hspace{1em} 1 : built before 1919
\hspace{1em} 2 : built between 1919 and 1948
\hspace{1em} 3 : built between 1949 and 1965
\hspace{1em} 4 : built between 1966 and 1977
\hspace{1em} 5 : built between 1978 and 1983
\hspace{1em} 6 : built after 1983
\textbf{wflkat} Floor space category (wfl categorized):
\hspace{1em} 1 : less than 50 sqm
\hspace{1em} 2 : between 51 sqm and 80 sqm
\hspace{1em} 3 : at least 81 sqm
\textbf{nmqm} Net rent per sqm.
\textbf{rooms} Number of rooms in household.
\textbf{nmkat} Net rent category (nm categorized):
\hspace{1em} 1 : less than 500 DM
\hspace{1em} 2 : between 500 DM and 675 DM
\hspace{1em} 3 : between 675 DM and 850 DM
\hspace{1em} 4 : between 850 DM and 1150 DM
\hspace{1em} 5 : at least 1150 DM
specClust

Address type:
1: bad
2: average
3: good

Residential type:
1: bad
2: average
3: good

Source


The data were converted to R format by <klaus.schliep@gmail.com>.

Examples

data(miete)
str(miete)

specClust  Spectral Clustering

Description

Spectral clustering based on k-nearest neighbor graph.

Usage

specClust(data, centers=NULL, nn = 7, method = "symmetric", gmax=NULL, ...)
## S3 method for class 'specClust'
plot(x, ...)

Arguments

data  Matrix or data frame.
centers  number of clusters to estimate, if NULL the number is chosen automatical.
nn  Number of neighbors considered.
method  Normalisation of the Laplacian ("none", "symmetric" or "random-walk").
gmax  maximal number of connected components.
x  an object of class specClust
...  Further arguments passed to or from other methods.
specClust

Details

specClust allows to estimate several popular spectral clustering algorithms, for an overview see von Luxburg (2007).

The Laplacian is constructed from a from nearest neighbors and there are several kernels available. The eigenvalues and eigenvectors are computed using the binding in igraph to arpack. This should ensure that this algorithm is also feasible for larger datasets as the the the distances used have dimension \( n \times m \), where \( n \) is the number of observations and \( m \) the number of nearest neighbors. The Laplacian is sparse and has roughly \( n \times m \) elements and only \( k \) eigenvectors are computed, where \( k \) is the number of centers.

Value

specClust returns a kmeans object or in case of \( k \) being a vector a list of kmeans objects.

Author(s)

Klaus P. Schliep <klaus.schliep@gmail.com>

References


See Also

kknn, arpack, kmeans

Examples

data(iris)
c1 <- specClust(iris[,1:4], 3, nn=5)
pcol <- as.character(as.numeric(iris$Species))
pairs(iris[1:4], pch = pcol, col = c("green", "red", "blue")[[c1$cluster]])
table(iris[,5], cl$cluster)
Training of kknn method via leave-one-out (`train.kknn`) or k-fold (`cv.kknn`) crossvalidation.

**Usage**

```r
train.kknn(formula, data, kmax = 11, ks = NULL, distance = 2, kernel = "optimal", ykernel = NULL, scale = TRUE, contrasts = c('unordered' = "contr.dummy", ordered = "contr.ordinal"), ...) 

cv.kknn(formula, data, kcv = 10, ...)
```

**Arguments**

- `formula`: A formula object.
- `data`: Matrix or data frame.
- `kmax`: Maximum number of k, if `ks` is not specified.
- `ks`: A vector specifying values of k. If not null, this takes precedence over `kmax`.
- `distance`: Parameter of Minkowski distance.
- `kernel`: Kernel to use. Possible choices are "rectangular" (which is standard unweighted knn), "triangular", "epanechnikov" (or beta(2,2)), "biweight" (or beta(3,3)), "triweight" (or beta(4,4)), "cos", "inv", "gaussian" and "optimal".
- `ykernel`: Window width of an y-kernel, especially for prediction of ordinal classes.
- `scale`: logical, scale variable to have equal sd.
- `contrasts`: A vector containing the 'unordered' and 'ordered' contrasts to use.
- `...`: Further arguments passed to or from other methods.
- `kcv`: Number of partitions for k-fold cross validation.

**Details**

- `train.kknn` performs leave-one-out crossvalidation and is computationally very efficient. `cv.kknn` performs k-fold crossvalidation and is generally slower and does not yet contain the test of different models yet.

**Value**

- `train.kknn` returns a list-object of class `train.kknn` including the components.
  - `MISCLASS`: Matrix of misclassification errors.
  - `MEAN.ABS`: Matrix of mean absolute errors.
  - `MEAN.SQU`: Matrix of mean squared errors.
  - `fitted.values`: List of predictions for all combinations of kernel and k.
best.parameters
List containing the best parameter value for kernel and k.

response
Type of response variable, one of continuous, nominal or ordinal.

distance
Parameter of Minkowski distance.

call
The matched call.

terms
The 'terms' object used.

Author(s)
Klaus P. Schliep <klaus.schliep@gmail.com>

References


See Also
kknn and simulation

Examples
library(kknn)
## Not run:
data(miete)
(train.con <- train.kknn(nmqm ~ wfl + bjkat + zh, data = miete, kmax = 25, kernel = c("rectangular", "triangular", "epanechnikov", "gaussian", "rank", "optimal")))
plot(train.con)
(train.ord <- train.kknn(wflkat ~ nm + bjkat + zh, miete, kmax = 25, kernel = c("rectangular", "triangular", "epanechnikov", "gaussian", "rank", "optimal")))
plot(train.ord)
(train.nom <- train.kknn(zh ~ wfl + bjkat + nmqm, miete, kmax = 25, kernel = c("rectangular", "triangular", "epanechnikov", "gaussian", "rank", "optimal")))
plot(train.nom)
## End(Not run)
data(glass)
glass <- glass[-1]
(fit.glass1 <- train.kknn(Type ~ ., glass, kmax = 15, kernel = c("triangular", "rectangular", "epanechnikov", "optimal"), distance = 1))
(fit.glass2 <- train.kknn(Type ~ ., glass, kmax = 15, kernel = c("triangular", "rectangular", "epanechnikov", "optimal"), distance = 2))
plot(fit.glass1)
plot(fit.glass2)
Index

* classif
  contr.dummy, 2
  kknn, 5
  kknn-deprecated, 7
  train.kknn, 12
* cluster
  specClust, 10
* datasets
  glass, 3
  ionosphere, 4
  miete, 8
* design
  contr.dummy, 2
* package
  kknn-package, 2

arpack, 11
contr.dummy, 2
contr.metric (contr.dummy), 2
contr.ordinal (contr.dummy), 2
contr.poly, 3
contr.sdif, 3
contrasts, 3
cv.kknn (train.kknn), 12

Defunct, 8
Deprecated, 8
glass, 3

ionosphere, 4

kknn, 5, 11, 13
kknn-deprecated, 7
kknn-package, 2
kmeans, 11
knn, 7
knn1, 7
miete, 8

plot.specClust (specClust), 10
plot.train.kknn (train.kknn), 12
predict.kknn (kknn), 5
predict.train.kknn (train.kknn), 12
print.kknn (kknn), 5
print.train.kknn (train.kknn), 12

simulation, 7, 13
simulation (kknn-deprecated), 7
specClust, 10
summary.kknn (kknn), 5
summary.train.kknn (train.kknn), 12

train.kknn, 7, 12