Package ‘discretization’

October 13, 2022

Type  Package
Title  Data Preprocessing, Discretization for Classification
Version  1.0-1.1
Date  2010-12-02
Author  HyunJi Kim
Maintainer  HyunJi Kim <polaris7867@gmail.com>
Description  A collection of supervised discretization algorithms. It can also be grouped in terms of top-down or bottom-up, implementing the discretization algorithms.
License  GPL
LazyLoad  yes
Repository  CRAN
Date/Publication  2022-06-09 09:13:40 UTC
NeedsCompilation  no

R topics documented:

  discretization-package .................................................. 2
  ameva ................................................................. 3
  cacc ................................................................. 4
  caim ................................................................. 5
  chi2 ................................................................. 6
  chiM ................................................................. 8
  chiSq ............................................................... 9
  cutIndex ............................................................ 10
  cutPoints ............................................................ 11
  disc.Topdown ......................................................... 11
  ent ................................................................. 12
  extendChi2 .......................................................... 13
  findBest ............................................................ 14
  incon ............................................................... 15
  insert ............................................................... 16
discretization-package

Data preprocessing, discretization for classification.

Description

This package is a collection of supervised discretization algorithms. It can also be grouped in terms of top-down or bottom-up, implementing the discretization algorithms.

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>discretization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>1.0-1</td>
</tr>
<tr>
<td>Date</td>
<td>2010-12-02</td>
</tr>
<tr>
<td>License</td>
<td>GPL</td>
</tr>
<tr>
<td>LazyLoad</td>
<td>yes</td>
</tr>
</tbody>
</table>

Author(s)

Maintainer: HyunJi Kim <polaris7867@gmail.com>

References


### ameva

**Auxiliary function for Ameva algorithm**

**Description**

This function is required to compute the ameva value for Ameva algorithm.

**Usage**

```r
ameva(tb)
```

**Arguments**

- `tb` a vector of observed frequencies, $k \times l$

**Details**

This function implements the Ameva criterion proposed in Gonzalez-Abril, Cuberos, Velasco and Ortega (2009) for Discretization. An autonomous discretization algorithm(Ameva) implements in \texttt{disc.Topdown(data,method=1)}. It uses a measure based on $\chi^2$ as the criterion for the optimal discretization which has the minimum number of discrete intervals and minimum loss of class variable interdependence. The algorithm finds local maximum values of Ameva criterion and a stopping criterion.

Ameva coefficient is defined as follows:

$$Ameva(k) = \frac{\chi^2(k)}{k \times (l - 1)}$$
for $k, l \geq 2$, $k$ is a number of intervals, $l$ is a number of classes.

This value calculates in contingency table between class variable and discrete interval, row matrix representing the class variable and each column of discrete interval.

**Value**

val numeric value of Ameva coefficient

**Author(s)**

HyunJi Kim <polaris7867@gmail.com>

**References**


**See Also**

disc.Topdown, topdown, insert, findBest and chiSq.

**Examples**

```r
#--Ameva criterion value
a=c(2,5,1,1,3,3)
m=matrix(a,ncol=3,byrow=TRUE)
ameva(m)
```

---

cacc Auxiliary function for CACC discretization algorithm

**Description**

This function is required to compute the cacc value for CACC discretization algorithm.

**Usage**

cacc(tb)

**Arguments**

tb a vector of observed frequencies
The Class-Attribute Contingency Coefficient (CACC) discretization algorithm implements in disc.Topdown(data, method=2).
The cacc value is defined as
\[ cacc = \sqrt{\frac{y}{y + M}} \]
for
\[ y = \frac{\chi^2}{\log(n)} \]

\( M \) is the total number of samples, \( n \) is a number of discretized intervals. This value calculates in contingency table between class variable and discrete interval, row matrix representing the class variable and each column of discrete interval.

Value

val numeric of cacc value

Author(s)

HyunJi Kim <polaris7867@gmail.com>

References


See Also
disc.Topdown, topdown, insert, findBest and chiSq.

Examples

```r
#----Calculating cacc value (Tsai, Lee, and Yang (2008))
a=c(3,0,3,0,6,0,0,3,0)
m=matrix(a,ncol=3,byrow=TRUE)
cacc(m)
```

caim

Auxiliary function for caim discretization algorithm

Description

This function is required to compute the CAIM value for CAIM discretization algorithm.

Usage

caim(tb)
Arguments

tb  a vector of observed frequencies

Details

The Class-Attribute Interdependence Maximization (CAIM) discretization algorithm implements in `disc.Topdown(data, method=1)`. The CAIM criterion measures the dependency between the class variable and the discretization variable for attribute, and is defined as:

\[
CAIM = \sum_{r=1}^{n} \frac{\max_r^2}{M_r} \frac{1}{n}
\]

for \( r = 1, 2, ..., n \), \( \max_r \) is the maximum value within the \( r \)th column of the quanta matrix. \( M_r \) is the total number of continuous values of attribute that are within the interval (Kurgan and Cios (2004)).

Author(s)

HyunJi Kim <polaris7867@gmail.com>

References


See Also

disc.Topdown, topdown, insert, findBest.

Examples

```r
#----Calculating caim value
a=c(3,0,3,0,6,0,0,3,0)
m=matrix(a,ncol=3,byrow=TRUE)
caim(m)
```

---

**Description**

This function performs Chi2 discretization algorithm. Chi2 algorithm automatically determines a proper Chi-square(\( \chi^2 \)) threshold that keeps the fidelity of the original numeric dataset.

**Usage**

`chi2(data, alp = 0.5, del = 0.05)`
**Arguments**

- `data` the dataset to be discretize
- `alp` significance level; $\alpha$
- `del` $Inconsistency(data) < \delta$, (Liu and Setiono(1995))

**Details**

The Chi2 algorithm is based on the $\chi^2$ statistic, and consists of two phases. In the first phase, it begins with a high significance level($\text{sigLevel}$), for all numeric attributes for discretization. Each attribute is sorted according to its values. Then the following is performed: **phase 1.** calculate the $\chi^2$ value for every pair of adjacent intervals (at the beginning, each pattern is put into its own interval that contains only one value of an attribute); **phase 2.** merge the pair of adjacent intervals with the lowest $\chi^2$ value. Merging continues until all pairs of intervals have $\chi^2$ values exceeding the parameter determined by $\text{sigLevel}$. The above process is repeated with a decreased $\text{sigLevel}$ until an inconsistency rate($\delta$), $\text{incon()}$, is exceeded in the discretized data(Liu and Setiono (1995)).

**Value**

- `cutp` list of cut-points for each variable
- `Disc.data` discretized data matrix

**Author(s)**

HyunJi Kim <polaris7867@gmail.com>

**References**


**See Also**

`value`, `incon` and `chiM`.

**Examples**

```r
data(iris)
#---cut-points
chi2(iris,0.5,0.05)$cutp

#--discretized dataset using Chi2 algorithm
chi2(iris,0.5,0.05)$Disc.data
```
ChiMerge discretization algorithm

**Description**

This function implements ChiMerge discretization algorithm.

**Usage**

```r
chiM(data, alpha = 0.05)
```

**Arguments**

- `data`: numeric data matrix to discretized dataset
- `alpha`: significance level; $\alpha$

**Details**

The ChiMerge algorithm follows the axis of bottom-up. It uses the $\chi^2$ statistic to determine if the relative class frequencies of adjacent intervals are distinctly different or if they are similar enough to justify merging them into a single interval (Kerber, R. (1992)).

**Value**

- `cutp`: list of cut-points for each variable
- `Disc.data`: discretized data matrix

**Author(s)**

HyunJi Kim <polaris7867@gmail.com>

**References**


**See Also**

`chiSq`, `value`.

**Examples**

```r
#--Discretization using the ChiMerge method
data(iris)
disc=chiM(iris,alpha=0.05)

#--cut-points
disc$cutp
```
chiSq

Auxiliary function for discretization using Chi-square statistic

Description

This function is required to perform the discretization based on Chi-square statistic (CACC, Ameva, ChiMerge, Chi2, Modified Chi2, Extended Chi2).

Usage

chiSq(tb)

Arguments

tb a vector of observed frequencies

Details

The formula for computing the $\chi^2$ value is

$$\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{k} \frac{(A_{ij} - E_{ij})^2}{E_{ij}}$$

$k =$ number of (no.) classes, $A_{ij} =$ no. patterns in the $i$th interval, $j$th class, $R_i =$ no. patterns in the $j$th class $= \sum_{j=1}^{k} A_{ij}$, $C_j =$ no. patterns in the $j$the class $= \sum_{i=1}^{2} A_{ij}$, $N =$ total no. patterns $= \sum_{i=1}^{2} R_{ij}$, $E_{ij} =$ expected frequency of $A_{ij} = R_i * C_j / N$. If either $R_i$ or $C_j$ is 0, $E_{ij}$ is set to 0.1. The degree of freedom of the $\chi^2$ statistic is on less the number of classes.

Value

val $\chi^2$ value

Author(s)

HyunJi Kim <polaris7867@gmail.com>

References


See Also

cacc, ameva, chiM, chi2, modChi2 and extendChi2.
Examples

```r
#----Calculate Chi-Square
b=c(2,4,1,2,5,3)
m=matrix(b,ncol=3)
chiSq(m)
chisq.test(m)$statistic
```

cutIndex

**Auxiliary function for the MDLP**

Description

This function is required to perform the Minimum Description Length Principle.\texttt{mdlp}

Usage

```r
cutIndex(x, y)
```

Arguments

- `x` a vector of numeric value
- `y` class variable vector

Details

This function computes the best cut index using entropy

Author(s)

HyunJi Kim <polaris7867@gmail.com>

See Also

cutPoints, ent, mergeCols, mdlStop, mylog, mdp.
cutPoints

**Auxiliary function for the MDLP**

**Description**
This function is required to perform the Minimum Description Length Principle.

**Usage**
cutPoints(x, y)

**Arguments**
- **x**: a vector of numeric value
- **y**: class variable vector

**Author(s)**
HyunJi Kim <polaris7867@gmail.com>

**See Also**
cutIndex, ent, mergeCols, mdlStop, mylog, mdlp.

disc.Topdown

**Top-down discretization**

**Description**
This function implements three top-down discretization algorithms (CAIM, CACC, Ameva).

**Usage**
disc.Topdown(data, method = 1)

**Arguments**
- **data**: numeric data matrix to discretized dataset

**Value**
- **cutp**: list of cut-points for each variable (minimum value, cut-points and maximum value)
- **Disc.data**: discretized data matrix
Author(s)
HyunJi Kim <polaris7867@gmail.com>

References


See Also
topdown, insert, findBest, findInterval, caim, cacc, ameva

Examples
```r
##---- CAIM discretization ----
##----cut-potins
cm=disc.Topdown(iris, method=1)
cm$cutp
##----discretized data matrix
cm$Disc.data

##---- CACC discretization----
disc.Topdown(iris, method=2)

##---- Ameva discretization ----
disc.Topdown(iris, method=3)
```

---

ent

*Auxiliary function for the MDLP*

Description
This function is required to perform the Minimum Description Length Principle.

Usage
```
et(y)
```

Arguments
```
y  class variable vector
```
extendChi2

**Author(s)**
HyunJi Kim <polaris7867@gmail.com>

**See Also**
cutPoints, ent, mergeCols, mdlStop, mylog, mdlp.

---

**extendChi2**  
*Discretization of Numeric Attributes using the Extended Chi2 algorithm*

**Description**
This function implements Extended Chi2 discretization algorithm.

**Usage**
```r
extendChi2(data, alp = 0.5)
```

**Arguments**
- `data`  
data matrix to discretized dataset
- `alp`  
significance level; $\alpha$

**Details**
In the extended Chi2 algorithm, inconsistency checking ($\text{InConCheck}(data) < \delta$) of the Chi2 algorithm is replaced by the lease upper bound $\xi(X_i())$ after each step of discretization ($\xi_{\text{discretized}} < \xi_{\text{original}}$). It uses as the stopping criterion.

**Value**
- `cutp`  
list of cut-points for each variable
- `Disc.data`  
discretized data matrix

**Author(s)**
HyunJi Kim <polaris7867@gmail.com>

**References**

**See Also**
chiM, Xi
findBest

Examples

data(iris)
ext=extendChi2(iris,0.5)
ext$cutp
ext$Disc.data

findBest

Auxiliary function for top-down discretization

Description

This function is required to perform the disc.Topdown() function.

Usage

findBest(x, y, bd, di, method)

Arguments

x a vector of numeric value
y class variable vector
bd current cut points
di candidate cut-points
method each method number indicates three top-down discretization. 1 for CAIM algorithm, 2 for CACC algorithm, 3 for Ameva algorithm.

Author(s)

HyunJi Kim <polaris7867@gmail.com>

See Also

topdown, insert and disc.Topdown.
Computing the inconsistency rate for Chi2 discretization algorithm

Description
This function computes the inconsistency rate of dataset.

Usage
incon(data)

Arguments
data dataset matrix

Details
The inconsistency rate of dataset is calculated as follows: (1) two instances are considered inconsistent if they match except for their class labels; (2) for all the matching instances (without considering their class labels), the inconsistency count is the number of the instances minus the largest number of instances of class labels; (3) the inconsistency rate is the sum of all the inconsistency counts divided by the total number of instances.

Value
inConRate the inconsistency rate of the dataset

Author(s)
HyunJi Kim <polaris7867@gmail.com>

References

See Also
chi2

Examples
```r
#---- Calculating Inconsistency ----
data(iris)
disiris=chiM(iris,alpha=0.05)$Disc.data
incon(disiris)
```
### insert

**Auxiliary function for Top-down discretization**

**Description**

This function is required to perform the `disc.Topdown()`.

**Usage**

```r
insert(x, a)
```

**Arguments**

- `x`: cut-point
- `a`: a vector of minimum, maximum value

**Author(s)**

HyunJi Kim <polaris7867@gmail.com>

**See Also**

`topdown`, `findBest` and `disc.Topdown`.

---

### LevCon

**Auxiliary function for the Modified Chi2 discretization algorithm**

**Description**

This function computes the level of consistency, is required to perform the Modified Chi2 discretization algorithm.

**Usage**

```r
LevCon(data)
```

**Arguments**

- `data`: discretized data matrix

**Value**

- `LevelConsis`: Level of Consistency value

**Author(s)**

HyunJi Kim <polaris7867@gmail.com>
References


See Also

modChi2

---

**mdlp**

*Discretization using the Minimum Description Length Principle (MDLP)*

**Description**

This function discretizes the continuous attributes of data matrix using entropy criterion with the Minimum Description Length as stopping rule.

**Usage**

`mdlp(data)`

**Arguments**

- `data` data matrix to be discretized dataset

**Details**

Minimum Discription Length Principle

**Value**

- `cutp` list of cut-points for each variable
- `Disc.data` discretized data matrix

**Author(s)**

HyunJi Kim <polaris7867@gmail.com>

**References**

mdlStop

See Also

cutIndex, cutPoints, ent, mergeCols, mdlStop, mylog.

Examples

data(iris)
mdlp(iris)$Disc.data

mdlStop

Auxiliary function for performing discretization using MDLP

Description

This function determines cut criterion based on Fayyad and Irani Criterion, is required to perform the minimum description length principle.

Usage

mdlStop(ci, y, entropy)

Arguments

ci cut index
y class variable
entropy this value is calculated by cutIndex()

Details

Minimum description Length Principle Criterion

Value

gain numeric value

Author(s)

HyunJi Kim <polaris7867@gmail.com>

References


See Also

cutPoints, ent, mergeCols, cutIndex, mylog, mdlp.
mergeCols

**Auxiliary function for performing discretization using MDLP**

**Description**

This function merges the columns having observation numbers equal to 0, required to perform the minimum description length principle.

**Usage**

```r
mergeCols(n, minimum = 2)
```

**Arguments**

- `n`: table, column: intervals, row: variables
- `minimum`: min # observations in col or row to merge

**Author(s)**

HyunJi Kim <polaris7867@gmail.com>

**See Also**

`cutPoints`, `ent`, `cutIndex`, `mdlStop`, `mylog`, `mdlp`, `modChi2`

---

**modChi2**

*Discretization of Numeric Attributes using the Modified Chi2 method*

**Description**

This function implements the Modified Chi2 discretization algorithm.

**Usage**

```r
modChi2(data, alp = 0.5)
```

**Arguments**

- `data`: numeric data matrix to discretized dataset
- `alp`: significance level, $\alpha$

**Details**

In the modified Chi2 algorithm, inconsistency checking($InConCheck(data) < \delta$) of the Chi2 algorithm is replaced by maintaining the level of consistency $L_c$ after each step of discretization ($L_{c\text{-discretized}} < L_{c\text{-original}}$), this inconsistency rate as the stopping criterion.
Value

cutp       list of cut-points for each variable
Disc.data  discretized data matrix

Author(s)

HyunJi Kim <polaris7867@gmail.com>

References


See Also

LevCon

Examples

data(iris)
modChi2(iris, alp=0.5)$Disc.data

---

mylog  *Auxiliary function for performing discretization using MDLP*

Description

This function is required to perform the minimum discription length principle, mdlp().

Usage

mylog(x)

Arguments

x  a vector of numeric value

Author(s)

HyunJi Kim <polaris7867@gmail.com>

References


See Also

mergeCols, ent, cutIndex, cutPoints, mdlStop and mdp.
topdown

**topdown**

*Auxiliary function for performing top-down discretization algorithm*

**Description**

This function is required to perform the `disc.Topdown()`.

**Usage**

```r
topdown(data, method = 1)
```

**Arguments**

- `data` numeric data matrix to discretized dataset

**Author(s)**

HyunJi Kim <polaris7867@gmail.com>

**References**


**See Also**

`insert`, `findBest` and `disc.Topdown`.

---

value

*Auxiliary function for performing the ChiMerge discretization*

**Description**

This function is called by ChiMerge diacretization function, `chM()`.

**Usage**

```r
value(i, data, alpha)
```
## Xi

### Arguments

- **i**
  - *i*th variable in data matrix to discretized
- **data**
  - numeric data matrix
- **alpha**
  - significance level; \( \alpha \)

### Value

- **cuts**
  - list of cut-points for any variable
- **disc**
  - discretized *i*th variable and data matrix of other variables

### Author(s)

HyunJi Kim <polaris7867@gmail.com>

### References


### See Also

*chiM*.

### Examples

```r
data(iris)
value(1, iris, 0.05)
```

---

**Description**

This function is the \( \xi \), required to perform the Extended Chi2 discretization algorithm.

**Usage**

\[ Xi(data) \]

### Arguments

- **data**
  - data matrix
Details

The following equality is used for calculating the least upper bound($\xi$) of the data set(Chao and Jyh-Hwa (2005)).

$$\xi(C, D) = \max(m_1, m_2)$$

where $C$ is the equivalence relation set, $D$ is the decision set, and $C^* = \{E_1, E_2, \ldots, E_n\}$ is the equivalence classes. $m_1 = 1 - \min\{c(E, D) | E \in C^* \text{ and } 0.5 < c(E, D)\}$, $m_2 = 1 - \max\{c(E, D) | E \in C^* \text{ and } c(E, D) < 0.5\}$.

$$c(E, D) = 1 - \frac{\text{card}(E \cap D)}{\text{card}(E)}$$

$\text{card}$ denotes set cardinality.

Value

Xi numeric value, $\xi$

Author(s)

HyunJi Kim <polaris7867@gmail.com>

References


See Also

extendChi2
Index

* package
  discretization-package, 2

ameva, 3, 9, 12

cacc, 4, 9, 12
caim, 5, 12
chi2, 6, 9, 15
chiM, 7, 8, 9, 13, 22
chiSq, 4, 5, 8, 9
cutIndex, 10, 11, 18–20
cutPoints, 10, 11, 13, 18–20
disc.Topdown, 4–6, 11, 14, 16, 21
discretization
  (discretization-package), 2
discretization-package, 2

ent, 10, 11, 12, 13, 18–20
extendChi2, 9, 13, 23

findBest, 4–6, 12, 14, 16, 21
findInterval, 12

incon, 7, 15
insert, 4–6, 12, 14, 16, 21

LevCon, 16, 20

mdl, 10, 11, 13, 17, 18–20
mdlStop, 10, 11, 13, 18, 19, 20
mergeCols, 10, 11, 13, 18, 19, 20
modChi2, 9, 17, 19
mylog, 10, 11, 13, 18, 19, 20

topdown, 4–6, 12, 14, 16, 21

value, 7, 8, 21

Xi, 13, 22

24