Package 'qVarSel'

February 20, 2015

Type Package

Title Variables Selection for Clustering and Classification

Version 1.0

Date 2014-05-27

Author Stefano Benati

Maintainer Stefano Benati <stefano.benati@unitn.it>

Description For a given data matrix A and cluster centers/prototypes collected in the matrix P, the functions described here select a subset of statistic variables Q that mostly explains/justifies P as prototypes. The functions are useful to reduce the data dimension for classification and to discard masking variables for clustering.

License GPL-2

Depends lpSolveAPI

Suggests clusterGeneration, mclust

Imports Rcpp (>= 0.11.0)

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-06-12 16:42:34

R topics documented:

qVarSel-package 2
PrtDist 3
qVarSelH 5
qVarSelLP 6

Index **9**

qVarSel-package *Selecting Variables for Clustering and Classification*

Description

For a given data matrix A and cluster centers/prototypes collected in the matrix P, the functions described here select a subset of variables Q that mostly explains/justifies P as prototipes. The functions are useful to reduce the dimension of the data for classification and to discard masking variables for clustering.

Details

Package:	qVarSel
Type:	Package
Version:	1.0
Date:	2014-05-27
License:	gpl-2

The package is useful to reduce the variable dimension for clustering. The example below shows the sequence of the operations. First, k-means can applied to the whole data sets, to calculate prototypes P. Then, distances between units U and P are calculated ans stored in a matrix D. Then, apply package subroutine q-VarSelH to select the most important variables. Apply EM optimization on data D for full clustering parameters estimation.

Author(s)

Stefano Benati

Maintainer: Stefano Benati <stefano.benati@unitn.it>

References

S. Benati, S. Garcia Quiles, J. Puerto "Optimization Methods to Select Variables for Clustering", Working Paper, Universidad de Sevilla, 2014

Examples

```
# Generate random cluster with masking variables

require(clusterGeneration)
tmp1 <- genRandomClust(numClust = 4, sepVal = 0.2, clustszind = 2,
                       rangeN = c( 100, 150 ),
                       numNonNoisy = 5, numNoisy = 5, numReplicate = 1,
                       fileName = "chk1")
a <- tmp1$datList$chk1_1
ass <- tmp1$memList$chk1_1
```

```
numunits <- length(ass)
noiseindex <- tmp1$noisyList$chk1_1
a <- scale(a)  #Standardzation for columns

# calculate data prototypes using k-means

sl2 <- kmeans(a, 4, iter.max = 200,
                    nstart = 10, algorithm = "L")
prototype = sl2$centers

# calculate distances between observations and prototypes
# Remark: d is a 3-dimensions matrix

d = PrtDist(a, prototype)

# Select 5 most representative variables, use 200 iterations

lsH <- qVarSelH(d, 5, maxit = 200)

# reduce the dimension of a

sq = 1:(dim(a)[2])
vrb = sq[lsH$x > 0.01]
a_reduced = a[ ,vrb]

# use the EM methodology for clustering on the reduced data

require(mclust)
sl1 <- Mclust(a_reduced, G = 4, modelName = "VVV")
```

PrtDist *Calculation of the distances between units and centers*

Description

Given a data set A, with a[i,k] be the measure of variable k on unit i, and a prototype set P, with p[j,k] be the measure of variable k on prototype j, the function calculate d[i,j,k], the i,j distance according variable k

Usage

```
PrtDist(a,
        p)
```

Arguments

a	Matrix with n rows (units) and m columns (variables)
p	Matrix with g rows (units) and m columns (variables)

Details

d[i,j,k] is the squared distance: d[i,j,k] = (a[i,k] - p[j,k])^2

Value

d: The 3-dimensional matrix of i,j,k distances

Note

The function has been written to simplify the code examples. It has been written as a R script with minimal vectorization, therefore it is not computationally much efficient. Moreover d(i,j,k) is the square of differences and users may prefer to employ other dissimilarity measures. In that case, they should better write their own function.

Author(s)

Stefano Benati

References

S. Benati, S. Garcia Quiles, J. Puerto "Optimization Methods to Select Variables for Clustering", Working Paper, Universidad de Sevilla, 2014

Examples

```
## Generate random 2 cluster with 20 masking variables
## and 10 true variables

require(clusterGeneration)
tmp1 <- genRandomClust(numClust = 2, sepVal = 0.2, clustszind = 2,
                         rangeN = c( 100, 150 ),
                         numNonNoisy = 10, numNoisy = 20, numReplicate = 1,
                         fileName = "chk1")
a <- tmp1$datList$chk1_1
a <- scale(a)  # Standardize for column


## Calculate two prototypes, using kmeans
y <- kmeans(a, 2, iter.max = 200, nstart = 10)
p = y$centers

## Calculate dist:
d <- PrtDist(a, p)
```

qVarSelH *Selection of Variables for Clustering or Data Dimension Reduction*

Description

The function implements the q-Vars heuristic described in the reference below. Given a 3-dimension matrix D, with d[i,j,k] being the distance between statistic unit i and prototype j measured through variable k, the function calculates the set of variables of cardinality q that mostly explains the prototypes.

Usage

```
qVarSelH(d,
         q,
         maxit = 100)
```

Arguments

d	A numeric 3-dimensional matrix where elements d(i,j,k) are the distances between observation i and cluster center/prototype j, that are measured through variable k.
q	A positive scalar, that is the number of variables to select
maxit	A positive scalar, that is the maximum number of iteration allowed

Details

The heuristic repeatedly selects a set of variables and then allocates units to prototypes, while a local optimum is reached. Random restart is used to continue the search until the maximum number of iteration is reached.

Value

obj	The value of the objective function
x	A 0-1 vector describing wheter variable k is selected: If x[k] = 1 then k is selected
ass	A vector of assignment of units to clusters: if ass[i] = j then unit i is assigned to the cluster represented by center/prototype j
bestit	The iteration in which the optimal solution is found

Note

The methodology is heuristic and some steps are random. It may be the case that different runs provide different solutions.

Author(s)

Stefano Benati

References

S. Benati, S. Garcia Quiles, J. Puerto "Optimization Methods to Select Variables for Clustering", Working Paper, Universidad de Sevilla, 2014

See Also

qVarSelLP

Examples

```
## Generate random 2 cluster with 20 masking variables
## and 10 true variables

require(clusterGeneration)
tmp1 <- genRandomClust(numClust = 2, sepVal = 0.2, clustszind = 2,
                        rangeN = c( 100, 150 ),
                        numNonNoisy = 10, numNoisy = 20, numReplicate = 1,
                        fileName = "chk1")
a <- tmp1$datList$chk1_1
a <- scale(a)  # Standardize for column


## Calculate two prototype, using kmeans
y <- kmeans(a, 2, iter.max = 200, nstart = 10)
p = y$centers

## Calculate dist:
d <- PrtDist(a, p)

## Calculate Best 10 variables:
lsH <- qVarSelH(d, 10, maxit = 200)
```

`qVarSelLP` *A Mixed Integer Linear Programming Formulation for the Variable Selection Problem*

Description

The function solve the mixed integer linear programming formulation of the variable selection problem.

Usage

```
qVarSelLP(d,
          q,
          binary = FALSE,
          write = FALSE)
```

Arguments

d	A 3-dimensional distance matrix, in which d[i,j,k] is the distance between unit i and prototype j, according to variable k.
q	The number of variables to selct.
binary	Set this value to TRUE if you wish to solve the problem with integer variables, or set it to FALSE if you just want to solve the continuous relaxation
write	Set this value to TRUE if you want that the optimization problem is exported in a file called qdistsel.lp

Details

The function solves the linear problem through the lpSolve solver available through the package lpSolveAPI. The linear programming formulation is the implementation of model F3 described in the paper below (without the median variables).

Value

status	The result of the optimization as output of the function solve() of library lpSolveAPI
obj	The value of the objective function
x	The value of the problem variables corresponding to the variable selection: x[j] = 1 means that variable j has been selected, 0 otherwise. If the continuous relaxation has been solved, the vector can contain fractional variables (most likely meaningless).

Note

The computational time to solve an integer programming problem can easily become exponential, therefore be carefull when set variable "binary" to TRUE, as you could wait days even to get the solution of a small scale problem. Even though the continuos version can contain fractional variables, comparing the objective functions of subroutines qVarSelH and qVarSelLP is a certificate of the solution quality.

Author(s)

Stefano Benati

References

S. Benati, S. Garcia Quiles, "A p-median model with distance selection", Working Paper, Universidad Carlos III de Madrid, 2012

See Also

lpSolveAPI

Examples

```
## Generate random cluster

 a <- rbind(cbind(rnorm(5, 0, 1), rnorm(5, 0, 3), rnorm(5, 0, 5)),
           cbind(rnorm(5, 5, 1), rnorm(5, 5, 3), rnorm(5, 5, 5) ))

 ## calculate data prototypes using k-means

 sl2 <- kmeans(a, 2, iter.max = 100, nstart = 2)
 p = sl2$centers

 ## calculate distances between observations and prototypes
 ## Remark: d is a 3-dimensions matrix

 d = PrtDist(a, p)

 ## Select 2 most representative variables, use heuristic

 lsH <- qVarSelH(d, 2, maxit = 200)

 ## Select 2 variables, use linear relaxation

 require(lpSolveAPI)
 lsC <- qVarSelLP(d, 2)


 ## check optimality

 if (abs(lsH$obj - lsC$obj) < 0.001)
     message = "Heuristic Solution is Optimal"
```

Index

∗Topic **classif**
PrtDist, 3
qVarSel-package, 2
qVarSelH, 5
qVarSelLP, 6
∗Topic **cluster**
PrtDist, 3
qVarSel-package, 2
qVarSelH, 5
qVarSelLP, 6
∗Topic **optimize**
PrtDist, 3
qVarSel-package, 2
qVarSelH, 5
qVarSelLP, 6

PrtDist, 3

qVarSel *(*qVarSel-package*)*, 2
qVarSel-package, 2
qVarSelH, 5
qVarSelLP, 6