Package 'qVarSel'

November 28, 2024

Type Package

Title Select Variables for Optimal Clustering

Version 1.1

Date 2024-11-24

Description Finding hidden clusters in structured data can be hindered
by the presence of masking variables. If not detected,
masking variables are used to calculate the overall similarities between units,
and therefore the cluster attribution is more imprecise.
The algorithm q-vars implements an optimization method to find the variables
that most separate units between clusters. In this way, masking variables can be
discarded from the data frame and the clustering is more accurate.
Tests can be found in Benati et al.(2017) <doi:10.1080/01605682.2017.1398206>.

License GPL (>= 2)

Imports Rcpp (>= 1.0.13), lpSolveAPI

Suggests mclust

LinkingTo Rcpp

RoxygenNote 7.3.2

NeedsCompilation yes

Author Stefano Benati [aut, cre] (<https://orcid.org/0000-0002-1928-5224>)

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

Repository CRAN

Date/Publication 2024-11-28 12:10:02 UTC

Contents

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 prototypes. The functions are useful to reduce the dimension of the data for classification as they discard masking variables for clustering.

Details

Package:	qVarSel
Type:	Package
Version:	1.1
Date:	2024-11-15
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 "Mixed Integer Linear Programming and heuristic methods for feature selection in clustering", Journal of the Operational Research Society, 69:9, (2018), pp. 1379-1395

Examples

```
# Simulated data with 100 units, 10 true variables,
# 10 masking variables, 2 hidden clusters

n1 = 50
n2 = 50
n_true_var = 10
n_mask_var = 10
g1 = matrix(rnorm(n1*n_true_var, 0, 1), ncol = n_true_var)
g2 = matrix(rnorm(n2*n_true_var, 2, 1), ncol = n_true_var)
```

```
m1 = matrix(runif((n1 + n2)*n_mask_var, min = 0, max = 5), ncol = n_mask_var)
a = cbind(rbind(g1, g2), m1)

## 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 10 most representative variables, use heuristic

lsH <- qVarSelH(d, 10, 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 effcient clustering on the reduced data

require(mclust)
sl1 <- Mclust(a_reduced, G = 2, 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 (prototypes) 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 "Mixed Integer Linear Programming and heuristic methods for feature selection in clustering", Journal of the Operational Research Society, 69:9, (2018), pp. 1379-1395

Examples

```
# Simulated data with 100 units, 10 true variables,
# 10 masking variables, 2 hidden clusters

n1 = 50
n2 = 50
n_true_var = 10
n_mask_var = 10
g1 = matrix(rnorm(n1*n_true_var, 0, 1), ncol = n_true_var)
g2 = matrix(rnorm(n2*n_true_var, 2, 1), ncol = n_true_var)
m1 = matrix(runif((n1 + n2)*n_mask_var, min = 0, max = 5), ncol = n_mask_var)
a = cbind(rbind(g1, g2), m1)

## 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)
```

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 "Mixed Integer Linear Programming and heuristic methods for feature selection in clustering", Journal of the Operational Research Society, 69:9, (2018), pp. 1379-1395

See Also

qVarSelLP

Examples

```
# Simulated data with 100 units, 10 true variables,
# 10 masking variables, 2 hidden clusters

n1 = 50
n2 = 50
n_true_var = 10
n_mask_var = 10
g1 = matrix(rnorm(n1*n_true_var, 0, 1), ncol = n_true_var)
g2 = matrix(rnorm(n2*n_true_var, 2, 1), ncol = n_true_var)
m1 = matrix(runif((n1 + n2)*n_mask_var, min = 0, max = 5), ncol = n_mask_var)
a = cbind(rbind(g1, g2), m1)

## 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 10 most representative variables, use heuristic

lsH <- qVarSelH(d, 10, maxit = 200)
```

---

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

Description

The function solves 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 P1 described in the paper below.

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, J. Puerto "Mixed Integer Linear Programming and heuristic methods for feature selection in clustering", Journal of the Operational Research Society, 69:9, (2018), pp. 1379-1395

See Also

lpSolveAPI

Examples

```
# Simulated data with 100 units, 10 true variables,
# 10 masking variables, 2 hidden clusters

n1 = 50
n2 = 50
n_true_var = 10
n_mask_var = 10
g1 = matrix(rnorm(n1*n_true_var, 0, 1), ncol = n_true_var)
g2 = matrix(rnorm(n2*n_true_var, 2, 1), ncol = n_true_var)
m1 = matrix(runif((n1 + n2)*n_mask_var, min = 0, max = 5), ncol = n_mask_var)
a = cbind(rbind(g1, g2), m1)

## 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 10 most representative variables, use heuristic

lsH <- qVarSelH(d, 10, maxit = 200)

## Select 10 variables, use linear relaxation

require(lpSolveAPI)
lsC <- qVarSelLP(d, 10)

## check optimality

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

Index

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

PrtDist, 3

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