Package ‘RECA’

February 19, 2015

Type Package

Title Relevant Component Analysis for Supervised Distance Metric Learning

Version 1.1

Date 2014-12-16

Author Nan Xiao <road2stat@gmail.com>

Maintainer Nan Xiao <road2stat@gmail.com>

Description Relevant Component Analysis (RCA) tries to find a linear transformation of the feature space such that the effect of irrelevant variability is reduced in the transformed space.

License GPL (>= 2)

URL https://github.com/road2stat/RECA

BugReports https://github.com/road2stat/RECA/issues

Suggests MASS

NeedsCompilation no

Repository CRAN

Date/Publication 2014-12-16 00:50:18

R topics documented:

RECA-package .............................................................. 2
rca ................................................................. 2

Index 6
The RECA package performs Relevant Component Analysis (RCA).

Package: RECA
Type: Package
Version: 1.1
License: GPL (>= 2)

Bug reports and feature requests could be sent to https://github.com/roadRstat/RECA/issues.

Nan Xiao <<road2stat@gmail.com>>

rca performs a relevant component analysis (RCA) for the given data. It takes a data set and a set of positive constraints as arguments and returns a linear transformation of the data space into better representation, alternatively, a Mahalanobis metric over the data space.

rca(x, chunks, useD = NULL)

x
n * d matrix or data frame of original data.

chunks
a vector of size N describing the chunklets: -1 in the i-th place says that point i does not belong to any chunklet; integer j in place i says that point i belongs to chunklet j; The chunklets indexes should be 1:number-of-chunklets.
optional. When not given, RCA is done in the original dimension and \( B \) is full rank. When \( \text{useD} \) is given, RCA is preceded by constraints based LDA which reduces the dimension to \( \text{useD} \). \( B \) in this case is of rank \( \text{useD} \).

Details

The new representation is known to be optimal in an information theoretic sense under a constraint of keeping equivalent data points close to each other.

The three returned argument are just different forms of the same output. If one is interested in a Mahalanobis metric over the original data space, the first argument is all she/he needs. If a transformation into another space (where one can use the Euclidean metric) is preferred, the second returned argument is sufficient. Using \( A \) and \( B \) is equivalent in the following sense:

\[
\text{if } y_1 = A \ast x_1, y_2 = A \ast y_2 \text{ then}
\]

\[
(x_2 - x_1)^T \ast B \ast (x_2 - x_1) = (y_2 - y_1)^T \ast (y_2 - y_1)
\]

Value

A list of the RCA results:

- \( \text{B} \): The RCA suggested Mahalanobis matrix. Distances between data points \( x_1, x_2 \) should be computed by \((x_2 - x_1)^T \ast B \ast (x_2 - x_1)\)
- \( \text{RCA} \): The RCA suggested transformation of the data. The data should be transformed by \( \text{RCA} \ast \text{data} \)
- \( \text{newX} \): The data after the RCA transformation. \( \text{newX} = \text{data} \ast \text{RCA} \)

Note

Note that any different sets of instances (chunklets), e.g. \{1, 3, 7\} and \{4, 6\}, might belong to the same class and might belong to different classes.

Author(s)

Nan Xiao <http://r2s.name>

References


Examples

```r
set.seed(42)
require(MASS)  # generate synthetic Gaussian data
k = 100L       # sample size of each class
n = 3L         # specify how many classes
N = k * n      # total sample size
x1 = mvrnorm(k, mu = c(-8, 6), matrix(c(15, 1, 2, 10), ncol = 2))
```
x2 = mvrnorm(k, mu = c(0, 0), matrix(c(15, 0, 2, 10), ncol = 2))
x3 = mvrnorm(k, mu = c(8, -6), matrix(c(15, 1, 2, 10), ncol = 2))
x = as.data.frame(rbind(x1, x2, x3))  # predictor
y = gl(n, k)  # response

# The fully labeled data set with 3 classes
plot(x[, 1L], x[, 2L], bg = c("#E41A1C", "#377EB8", "#4DAF4A"[y],
    pch = rep(c(22, 21, 25), each = k))
Sys.sleep(2)

# Same data unlabeled; clearly the class structure is less evident
plot(x[, 1L], x[, 2L])
Sys.sleep(2)

# Manually generating synthetic chunklets
chunk1 = sample(1L:100L, 3L)
chunk2 = sample(1L:100L, 3L)
chunk3 = sample(1L:100L, 3L)
chunk4 = sample(1L:100L, 3L)
chunk5 = sample(1L:100L, 3L)
chunk6 = sample(1L:100L, 3L)
chunk7 = sample(101L:200L, 3L)
chunk8 = sample(101L:200L, 3L)
chunk9 = sample(101L:200L, 3L)
chunk10 = sample(101L:200L, 3L)
chunk11 = sample(101L:200L, 3L)
chunk12 = sample(101L:200L, 3L)
chunk13 = sample(101L:200L, 3L)
chunk14 = sample(101L:200L, 3L)
chunk15 = sample(201L:300L, 3L)
chunk16 = sample(201L:300L, 3L)
chunk17 = sample(201L:300L, 3L)
chunk18 = sample(201L:300L, 3L)
chunk19 = sample(201L:300L, 3L)
chunk20 = sample(201L:300L, 3L)
chks = x[c(chunk1, chunk2, chunk3, chunk4, chunk5,
    chunk6, chunk7, chunk8, chunk9, chunk10,
    chunk11, chunk12, chunk13, chunk14, chunk15,
    chunk16, chunk17, chunk18, chunk19, chunk20), ]
chunks = list(chunk1, chunk2, chunk3, chunk4, chunk5,
    chunk6, chunk7, chunk8, chunk9, chunk10,
    chunk11, chunk12, chunk13, chunk14, chunk15,
    chunk16, chunk17, chunk18, chunk19, chunk20)

# Make 'chunklet' vector to feed the chunks argument
chunkvec = rep(-1L, nrow(x))
for ( i in 1L:length(chunks) ) {
    for ( j in 1L:length(chunks[[i]]) ) {
        chunkvec[chunks[[i]][j]] = i
    }
}

# The chunklets provided to the RCA algorithm
plot(chks[, 1L], chks[, 2L], col = rep(1L:20L, each = 3L),
    pch = rep(0L:19L, each = 3L))
Sys.sleep(2)

# The RCA suggested transformation of the data
rca(x, chunksvec)$RCA

# The RCA suggested Mahalanobis matrix
rca(x, chunksvec)$B

# Whitening transformation applied to the chunklets
chkTransformed = as.matrix(chks) %*% rca(x, chunksvec)$RCA
plot(chkTransformed[, 1L], chkTransformed[, 2L],
    col = rep(1L:20L, each = 3L),
    pch = rep(0L:19L, each = 3L))
Sys.sleep(2)

# The origin data after applying the RCA transformation
xnew = rca(x, chunksvec)$newX
plot(xnew[, 1L], xnew[, 2L],
    bg = c("#E41A1C", "#377EB8", "#4DAF4A")%gl(n, k),
    pch = c(rep(22, k), rep(21, k), rep(25, k)))
Index

rca, 2
RECA-package, 2