Package: dml (via r-universe)

August 30, 2024

Type Package

Title Distance Metric Learning in R

Version 1.1.0.9001

Maintainer Yuan Tang <`terrytangyuan@gmail.com`>

Description State-of-the-art algorithms for distance metric learning, including global and local methods such as Relevant Component Analysis, Discriminative Component Analysis, Local Fisher Discriminant Analysis, etc. These distance metric learning methods are widely applied in feature extraction, dimensionality reduction, clustering, classification, information retrieval, and computer vision problems.

License MIT + file LICENSE

URL `https://github.com/terrytangyuan/dml`

Encoding UTF-8

Imports lfda

Suggests MASS, scatterplot3d, lintr, testthat

BugReports `https://github.com/terrytangyuan/dml/issues`

RoxygenNote 6.1.0

Repository https://terrytangyuan.r-universe.dev

RemoteUrl https://github.com/terrytangyuan/dml

RemoteRef HEAD

RemoteSha 68558d8b81d5cbb12ec7fe83642186f1c11576c4

Contents

dml-package *Distance Metric Learning in R*

Description

Solving distance metric learning problems with R.

Details

Package:	dml
Type:	Package
License:	MIT

Author(s)

Yuan Tang <<`terrytangyuan@gmail.com`>> Tao Gao <<`joegaotao@gmail.com`>> Nan Xiao <<`me@nanx.me`>>

dca *Discriminative Component Analysis*

Description

Performs discriminative component analysis on the given data.

Usage

```
dca(data, chunks, neglinks, useD = NULL)
```

Arguments

`data`	`n * d` data matrix. `n` is the number of data points, `d` is the dimension of the data. Each data point is a row in the matrix.
`chunks`	length `n` vector describing the chunklets: `-1` in the `i` th place means point `i` doesn't belong to any chunklet; integer `j` in place `i` means point `i` belongs to chunklet j. The chunklets indexes should be 1:(number of chunklets).
`neglinks`	`s * s` symmetric matrix describing the negative relationship between all the `s` chunklets. For the element $neglinks_{ij}$: $neglinks_{ij} = 1$ means chunklet `i` and chunklet j have negative constraint(s); $neglinks_{ij} = 0$ means chunklet `i` and chunklet j don't have negative constraints or we don't have information about that.
`useD`	Integer. Optional. When not given, DCA is done in the original dimension and B is full rank. When useD is given, DCA is preceded by constraints based LDA which reduces the dimension to useD. B in this case is of rank useD.

Value

list of the DCA results:

`B`	DCA suggested Mahalanobis matrix
`DCA`	DCA suggested transformation of the data. The dimension is (original data dimension) * (useD)
`newData`	DCA transformed data

For every two original data points (x1, x2) in newData (y1, y2):

$(x2 - x1)' * B * (x2 - x1) = ||(x2 - x1) * A||^2 = ||y2 - y1||^2$

References

Steven C.H. Hoi, W. Liu, M.R. Lyu and W.Y. Ma (2006). Learning Distance Metrics with Contextual Constraints for Image Retrieval. *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR2006).*

Examples

```
## Not run:
# generate synthetic Gaussian data
library("MASS")

k <- 100 # sample size of each class
n <- 3 # specify how many class
N <- k * n # total sample number

set.seed(123)
x1 <- mvrnorm(k, mu = c(-10, 6), matrix(c(10, 4, 4, 10), ncol = 2))
x2 <- mvrnorm(k, mu = c(0, 0), matrix(c(10, 4, 4, 10), ncol = 2))
x3 <- mvrnorm(k, mu = c(10, -6), matrix(c(10, 4, 4, 10), ncol = 2))
data <- as.data.frame(rbind(x1, x2, x3))
```

```
# The fully labeled data set with 3 classes
plot(data$V1, data$V2,
     bg = c("#E41A1C", "#377EB8", "#4DAF4A")[gl(n, k)],
     pch = c(rep(22, k), rep(21, k), rep(25, k))
)
Sys.sleep(3)

# The same data unlabeled; clearly the class structure is less evident
plot(data$V1, data$V2)
Sys.sleep(3)

chunk1 <- sample(1:100, 5)
chunk2 <- sample(setdiff(1:100, chunk1), 5)
chunk3 <- sample(101:200, 5)
chunk4 <- sample(setdiff(101:200, chunk3), 5)
chunk5 <- sample(201:300, 5)
chks <- list(chunk1, chunk2, chunk3, chunk4, chunk5)
chunks <- rep(-1, 300)

# positive samples in the chunks
for (i in 1:5) {
  for (j in chks[[i]]) {
    chunks[j] <- i
  }
}

# define the negative constrains between chunks
neglinks <- matrix(c(
  0, 0, 1, 1, 1,
  0, 0, 1, 1, 1,
  1, 1, 0, 0, 0,
  1, 1, 0, 0, 1,
  1, 1, 1, 1, 0
),
ncol = 5, byrow = TRUE
)

dcaData <- dca(data = data, chunks = chunks, neglinks = neglinks)$newData

# plot DCA transformed data
plot(dcaData[, 1], dcaData[, 2],
     bg = c("#E41A1C", "#377EB8", "#4DAF4A")[gl(n, k)],
     pch = c(rep(22, k), rep(21, k), rep(25, k)),
     xlim = c(-15, 15), ylim = c(-15, 15)
)
## End(Not run)
```

`GdmDiag` *Global Distance Metric Learning*

Description

Performs Global Distance Metric Learning (GDM) on the given data, learning a diagonal matrix.

Usage

```
GdmDiag(data, simi, dism, C0 = 1, threshold = 0.001)
```

Arguments

`data`	`n * d` data matrix. `n` is the number of data points, `d` is the dimension of the data. Each data point is a row in the matrix.
`simi`	`n * 2` matrix describing the similar constrains. Each row of matrix is serial number of a similar pair in the original data. For example, pair(1, 3) represents the first observation is similar the 3th observation in the original data.
`dism`	`n * 2` matrix describing the dissimilar constrains as `simi`. Each row of matrix is serial number of a dissimilar pair in the original data.
`C0`	numeric, the bound of similar constrains.
`threshold`	numeric, the threshold of stoping the learning iteration.

Value

list of the GdmDiag results:

`newData`	GdmDiag transformed data
`diagonalA`	suggested Mahalanobis matrix
`dmlA`	matrix to transform data, square root of diagonalA
`error`	the precision of obtained distance metric by Newton-Raphson optimization

For every two original data points (x1, x2) in newData (y1, y2):

$(x2 - x1)' * A * (x2 - x1) = ||(x2 - x1) * B||^2 = ||y2 - y1||^2$

Note

Be sure to check whether the dimension of original data and constrains' format are valid for the function.

Author(s)

Tao Gao <`<joegaotao@gmail.com>`>

References

Steven C.H. Hoi, W. Liu, M.R. Lyu and W.Y. Ma (2003). Distance metric learning, with application to clustering with side-information.

Examples

```
## Not run:
library("MASS")
library("scatterplot3d")

set.seed(602)

# generate simulated Gaussian data
k = 100
m <- matrix(c(1, 0.5, 1, 0.5, 2, -1, 1, -1, 3), nrow =3, byrow = T)
x1 <- mvrnorm(k, mu = c(1, 1, 1), Sigma = m)
x2 <- mvrnorm(k, mu = c(-1, 0, 0), Sigma = m)
data <- rbind(x1, x2)

# define similar constrains
simi <- rbind(t(combn(1:k, 2)), t(combn((k+1):(2*k), 2)))

temp <-  as.data.frame(t(simi))
tol <- as.data.frame(combn(1:(2*k), 2))

# define disimilar constrains
dism <- t(as.matrix(tol[!tol %in% simi]))

# transform data using GdmDiag
result <- GdmDiag(data, simi, dism)
newData <- result$newData
# plot original data
color <- gl(2, k, labels = c("red", "blue"))
par(mfrow = c(2, 1), mar = rep(0, 4) + 0.1)
scatterplot3d(data, color = color, cex.symbols = 0.6,
     xlim = range(data[, 1], newData[, 1]),
     ylim = range(data[, 2], newData[, 2]),
     zlim = range(data[, 3], newData[, 3]),
     main = "Original Data")
# plot GdmDiag transformed data
scatterplot3d(newData, color = color, cex.symbols = 0.6,
     xlim = range(data[, 1], newData[, 1]),
     ylim = range(data[, 2], newData[, 2]),
     zlim = range(data[, 3], newData[, 3]),
     main = "Transformed Data")

## End(Not run)
```

`GdmFull` *Global Distance Metric Learning*

Description

Performs Global Distance Metric Learning (GDM) on the given data, learning a full matrix.

Usage

```
GdmFull(data, simi, dism, maxiter = 100)
```

Arguments

`data`	`n * d` data matrix. `n` is the number of data points, `d` is the dimension of the data. Each data point is a row in the matrix.
`simi`	`n * 2` matrix describing the similar constrains. Each row of matrix is serial number of a similar pair in the original data. For example, pair(1, 3) represents the first observation is similar the 3th observation in the original data.
`dism`	`n * 2` matrix describing the dissimilar constrains as `simi`. Each row of matrix is serial number of a dissimilar pair in the original data.
`maxiter`	numeric, the number of iteration.

Value

list of the GdmDiag results:

`newData`	GdmDiag transformed data
`fullA`	suggested Mahalanobis matrix
`dmlA`	matrix to transform data, square root of diagonalA
`converged`	whether the iteration-projection optimization is converged or not

For every two original data points (x1, x2) in newData (y1, y2):

$(x2 - x1)' * A * (x2 - x1) = ||(x2 - x1) * B||^2 = ||y2 - y1||^2$

Note

Be sure to check whether the dimension of original data and constrains' format are valid for the function.

Author(s)

Tao Gao <`<joegaotao@gmail.com>`>

References

Steven C.H. Hoi, W. Liu, M.R. Lyu and W.Y. Ma (2003). Distance metric learning, with application to clustering with side-information.

Examples

```
## Not run:
set.seed(123)
library("MASS")
library("scatterplot3d")

# generate simulated Gaussian data
```

```
k = 100
m <- matrix(c(1, 0.5, 1, 0.5, 2, -1, 1, -1, 3), nrow =3, byrow = T)
x1 <- mvrnorm(k, mu = c(1, 1, 1), Sigma = m)
x2 <- mvrnorm(k, mu = c(-1, 0, 0), Sigma = m)
data <- rbind(x1, x2)

# define similar constrains
simi <- rbind(t(combn(1:k, 2)), t(combn((k+1):(2*k), 2)))

temp <-  as.data.frame(t(simi))
tol <- as.data.frame(combn(1:(2*k), 2))

# define disimilar constrains
dism <- t(as.matrix(tol[!tol %in% simi]))

# transform data using GdmFull
result <- GdmFull(data, simi, dism)
newData <- result$newData
# plot original data
color <- gl(2, k, labels = c("red", "blue"))
par(mfrow = c(2, 1), mar = rep(0, 4) + 0.1)
scatterplot3d(data, color = color, cex.symbols = 0.6,
     xlim = range(data[, 1], newData[, 1]),
     ylim = range(data[, 2], newData[, 2]),
     zlim = range(data[, 3], newData[, 3]),
     main = "Original Data")
# plot GdmFull transformed data
scatterplot3d(newData, color = color, cex.symbols = 0.6,
     xlim = range(data[, 1], newData[, 1]),
     ylim = range(data[, 2], newData[, 2]),
     zlim = range(data[, 3], newData[, 3]),
     main = "Transformed Data")

## End(Not run)
```

print.dca *Print an dca object*

Description

Print an dca object

Usage

```
## S3 method for class 'dca'
print(x, ...)
```

Arguments

`x`	The result from the dca function.
`...`	ignored

`print.GdmDiag` *Print an GdmDiag object*

Description

Print an GdmDiag object

Usage

```
## S3 method for class 'GdmDiag'
print(x, ...)
```

Arguments

`x`	The result from GdmDiag function.
`...`	ignored

`print.gmdf` *Print a gmdf object*

Description

Print a gmdf object

Usage

```
## S3 method for class 'gmdf'
print(x, ...)
```

Arguments

`x`	The result from gmdf function, which contains TK
`...`	ignored

`print.rca` *Print an rca object*

Description

Print an rca object

Usage

```
## S3 method for class 'rca'
print(x, ...)
```

Arguments

`x`	The result from rca function, which contains mahalanobis metric, whitening transformation matrix, and transformed data
`...`	ignored

`rca` *Relevant Component Analysis*

Description

Performs relevant component analysis on the given data.

Usage

```
rca(x, chunks, useD = NULL)
```

Arguments

`x`	matrix or data frame of original data. Each row is a feature vector of a data instance.
`chunks`	list of `k` numerical vectors. Each vector represents a chunklet, the elements in the vectors indicate where the samples locate in `x`. See examples for more information.
`useD`	optional. When not given, RCA is done in the original dimension and `B` is full rank. When `useD` is given, RCA is preceded by constraints based LDA which reduces the dimension to `useD`. `B` in this case is of rank `useD`.

Details

The RCA function 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.

Relevant component analysis consists of three steps:

1. locate the test point
2. compute the distances between the test points
3. find k shortest distances and the bla

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.

Value

list of the RCA results:

`B`	The RCA suggested Mahalanobis matrix. Distances between data points x1, x2 should be computed by (x2 - x1)' * B * (x2 - x1)
`A`	The RCA suggested transformation of the data. The data should be transformed by A * data
`newX`	The data after the RCA transformation (A). newData = A * data

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:

if y1 = A * x1, y2 = A * y2 then (x2 - x1)' * B * (x2 - x1) = (y2 - y1)' * (y2 - y1)

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 <`https://nanx.me`>

References

Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall (2003). Learning Distance Functions using Equivalence Relations. *Proceedings of 20th International Conference on Machine Learning (ICML2003).*

See Also

See `dca` for exploiting negative constrains.

Examples

```
## Not run:
library("MASS") # generate synthetic multivariate normal data
set.seed(42)
k <- 100L # sample size of each class
n <- 3L # specify how many classes
N <- k * n # total sample size
x1 <- mvrnorm(k, mu = c(-16, 8), matrix(c(15, 1, 2, 10), ncol = 2))
x2 <- mvrnorm(k, mu = c(0, 0), matrix(c(15, 1, 2, 10), ncol = 2))
x3 <- mvrnorm(k, mu = c(16, -8), matrix(c(15, 1, 2, 10), ncol = 2))
x <- as.data.frame(rbind(x1, x2, x3)) # predictors
y <- gl(n, k) # response

# fully labeled data set with 3 classes
# need to use a line in 2D to classify
plot(x[, 1L], x[, 2L],
  bg = c("#E41A1C", "#377EB8", "#4DAF4A")[y],
  pch = rep(c(22, 21, 25), each = k)
)
abline(a = -10, b = 1, lty = 2)
abline(a = 12, b = 1, lty = 2)

# generate synthetic chunklets
chunks <- vector("list", 300)
for (i in 1:100) chunks[[i]] <- sample(1L:100L, 10L)
for (i in 101:200) chunks[[i]] <- sample(101L:200L, 10L)
for (i in 201:300) chunks[[i]] <- sample(201L:300L, 10L)

chks <- x[unlist(chunks), ]

# make "chunklet" vector to feed the chunks argument
chunksvec <- rep(-1L, nrow(x))
for (i in 1L:length(chunks)) {
  for (j in 1L:length(chunks[[i]])) {
    chunksvec[chunks[[i]][j]] <- i
  }
}

# relevant component analysis
rcs <- rca(x, chunksvec)

# learned transformation of the data
rcs$A

# learned Mahalanobis distance metric
rcs$B

# whitening transformation applied to the chunklets
chkTransformed <- as.matrix(chks) %*% rcs$A

# original data after applying RCA transformation
# easier to classify - using only horizontal lines
```

```
xnew <- rcs$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))
)
abline(a = -15, b = 0, lty = 2)
abline(a = 16, b = 0, lty = 2)

## End(Not run)
```

Index