Package ‘lfda’

January 5, 2017

Type Package
Title Local Fisher Discriminant Analysis
Version 1.1.2
Date 2017-01-05
Author Yuan Tang. Contributions from Nan Xiao and Zachary Deane-Mayer.
URL https://github.com/terrytangyuan/lfda
BugReports https://github.com/terrytangyuan/lfda/issues
Maintainer Yuan Tang <terrytangyuan@gmail.com>
License MIT + file LICENSE
Description Functions for performing and visualizing Local Fisher Discriminant Analysis (LFDA), Kernel Fisher Discriminant Analysis (KLFDA), and Semi-supervised Local Fisher Discriminant Analysis (SELF).
Depends R (>= 3.1.0)
Imports plyr, grDevices, rARPACK
Suggests testthat, rgl
RoxygenNote 5.0.1
NeedsCompilation no
Repository CRAN
Date/Publication 2017-01-05 11:34:03

R topics documented:

Cols ......................................................... 2
getAffinityMatrix .......................................... 2
getMetricOfType ........................................... 3
klfda .................................................... 3
kmatrixGauss .............................................. 5
lfda ...................................................... 6
plot.lfda ................................................ 7
predict.lfda .............................................. 8
### assignColors

This function assigns a color to each distinct value in the given vector.

**Usage**

assignColors(vec)

**Arguments**

- **vec**: The vector where each distinct value will be assigned a color.

**Value**

The colors for each element in the given vector.

### getAffinityMatrix

This function returns an affinity matrix within knn-nearest neighbors from the distance matrix.

**Usage**

getAffinityMatrix(distance2, knn, nc)

**Arguments**

- **distance2**: The distance matrix for each observation
- **knn**: The number of nearest neighbors
- **nc**: The number of observations for data in this class

**Value**

an affinity matrix - the larger the element in the matrix, the closer two data points are
### getMetricOfType

**Get Requested Type of Transforming Metric**

**Description**

This function returns the requested type of transforming metric.

**Usage**

```r
getMetricOfType(metric, eigVec, eigVal, total)
```

**Arguments**

- `metric`: The type of metric to be requested
- `eigVec`: The eigenvectors of the problem
- `eigVal`: The eigenvalues of the problem
- `total`: The number of total rows to be used for weighting denominator

**Value**

The transformation metric in requested type

---

### klfda

**Kernel Local Fisher Discriminant Analysis for Supervised Dimensionality Reduction**

**Description**

Performs kernel local fisher discriminant analysis on the given data, which is the non-linear version of LFDA (see details `lfda`).

**Usage**

```r
klfda(k, y, r, metric = c("weighted", "orthonormalized", "plain"), knn = 6, reg = 0.001)
```

**Arguments**

- `k`: n x n kernel matrix. Result of the `kmatrixGauss` function. n is the number of samples
- `y`: n dimensional vector of class labels
- `r`: dimensionality of reduced space (default: d)
- `metric`: type of metric in the embedding space (default: 'weighted') 'weighted' — weighted eigenvectors 'orthonormalized' — orthonormalized 'plain' — raw eigenvectors
- `knn`: parameter used in local scaling method (default: 6)
- `reg`: regularization parameter (default: 0.001)
Value

- list of the LFDA results:
  - $\mathbf{t}$: d x r transformation matrix ($Z = \mathbf{t}(\mathbf{T}) \times X$)
  - $\mathbf{Z}$: r x n matrix of dimensionality reduced samples

Author(s)

Yuan Tang

References


Original Matlab Implementation: [http://www.ms.k.u-tokyo.ac.jp/software.html#LFDA](http://www.ms.k.u-tokyo.ac.jp/software.html#LFDA)

See Also

- See `lfda` for the linear version.

Examples

```r
## Not run:
## example without dimension reduction
k <- kmatrixGauss(x = trainData[, c(1)])
y <- trainData[, c(1)]
r <- 20 # dimensionality of reduced space. Here no dimension reduction is performed
result <- klfda(k, y, r, metric = "plain")
transformedMat <- result$Z # transformed training data
metric.train <- as.data.frame(cbind(trainData[, c(1)], transformedMat))
colnames(metric.train) <- colnames(trainData)

## example with dimension reduction
k <- kmatrixGauss(x = trainData[, c(1)])
y <- trainData[, c(1)]
r <- 3 # dimensionality of reduced space
result <- klfda(k, y, r, metric = "plain")
transformMat <- result$T # transforming matrix - distance metric

# transformed training data with Style
transformedMat <- result$Z # transformed training data
metric.train <- as.data.frame(cbind(trainData[, c(1)], transformedMat))
colnames(metric.train)[c(1)] <- "Style"

# transformed testing data with Style (unfinished)
metric.test <- kmatrixGauss(x = testData[, c(1)])
```
kmatrixGauss

Gaussian Kernel Computation (Particularly used in Kernel Local Fisher Discriminant Analysis)

Description

Gaussian kernel computation for klfda, which maps the original data space to non-linear and higher dimensions.

Usage

kmatrixGauss(x, sigma = 1)

Arguments

x n x d matrix of original samples. n is the number of samples.
sigma dimensionality of reduced space. (default: 1)

Value

K n x n kernel matrix. n is the number of samples.

Author(s)

Yuan Tang

References


https://shapeofdata.wordpress.com/2013/07/23/gaussian-kernels/

See Also

See klfda for the computation of kernel local fisher discriminant analysis

kn <- as.matrix(testData[, -1]) # transformMat
metric.test <- as.data.frame(cbind(testData[, 1], metric.test))
colnames(metric.test)[1] <- "Style"

## End(Not run)
Local Fisher Discriminant Analysis for Supervised Dimensionality Reduction

Description

Performs local fisher discriminant analysis (LFDA) on the given data.

Usage

\texttt{lfda(x, y, r, metric = c("orthonormalized", "plain", "weighted"), knn = 5)}

Arguments

\begin{itemize}
  \item \texttt{x} \quad n \times d matrix of original samples. \( n \) is the number of samples.
  \item \texttt{y} \quad length \( n \) vector of class labels
  \item \texttt{r} \quad dimensionality of reduced space (default: \( d \))
  \item \texttt{metric} \quad type of metric in the embedding space (no default) ‘weighted’ — weighted eigenvectors ‘orthonormalized’ — orthonormalized ‘plain’ — raw eigenvectors
  \item \texttt{knn} \quad parameter used in local scaling method (default: 5)
\end{itemize}

Details

LFDA is a method for linear dimensionality reduction that maximizes between-class scatter and minimizes within-class scatter while at the same time maintain the local structure of the data so that multimodal data can be embedded appropriately. Its limitation is that it only looks for linear boundaries between clusters. In this case, a non-linear version called kernel LFDA will be used instead. Three metric types can be used if needed.

Value

list of the LFDA results:

\begin{itemize}
  \item \texttt{T} \quad d \times r transformation matrix (\( Z = x \times T \))
  \item \texttt{Z} \quad n \times r matrix of dimensionality reduced samples
\end{itemize}

Author(s)

Yuan Tang
References


See Also

See `klfda` for the kernelized variant of LFDA (Kernel LFDA).

Examples

```r
## Not run:
## example without dimension reduction
k <- trainData[, -1]
y <- trainData[, 1]
r <- 26 # dimensionality of reduced space. Here no dimension reduction is performed
result <- lfdak(k, y, r, metric = "plain")
transformedMat <- result$Z # transformed training data
metric.train <- as.data.frame(cbind(trainData[, 1], transformedMat))
colnames(metric.train) <- colnames(trainData)

## example with dimension reduction
k <- trainData[, -1]
y <- trainData[, 1]
r <- 3 # dimensionality of reduced space
result <- lfdak(k, y, r, metric = "weighted")
transformMat <- result$T # transforming matrix - distance metric

# transformed training data with Style
transformedMat <- result$Z # transformed training data
metric.train <- as.data.frame(cbind(trainData[, 1], transformedMat))
colnames(metric.train)[1] <- "Style"

# transformed testing data with Style
metric.test <- as.matrix(testData[, -1]) %*% transformMat
metric.test <- as.data.frame(cbind(testData[, 1], metric.test))
colnames(metric.test)[1] <- "Style"

## End(Not run)
```
Description

This function plots 3 dimensions of the lfda/klfda result.

Usage

```r
## S3 method for class 'lfda'
plot(x, labels, cleanText = FALSE, ...)
```

Arguments

- `x`: The lfda/klfda result.
- `labels`: A list of class labels used for lfda/klfda training.
- `cleanText`: A boolean value to specify whether to make the labels in the plot cleaner (default: FALSE)
- `...`: Additional arguments

See Also

See `lfda` and `klfda` for the metric learning method used for this visualization.

---

**predict.lfda**

LFDA Transformation/Prediction on New Data

Description

This function transforms a data set, usually a testing set, using the trained LFDA metric.

Usage

```r
## S3 method for class 'lfda'
predict(object, newdata = NULL, type = "raw", ...)
```

Arguments

- `object`: The result from lfda function, which contains a transformed data and a transforming matrix that can be used for transforming testing set
- `newdata`: The data to be transformed
- `type`: The output type, in this case it defaults to "raw" since the output is a matrix
- `...`: Additional arguments

Value

the transformed matrix

Author(s)

Yuan Tang
print.lfda

Print an lfda object

Description

Print an lfda object

Usage

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

Arguments

x The result from lfda function, which contains a transformed data and a transforming
...
... ignored

repmat

Matlab-Syntaxed Repmat

Description

This function mimics the behavior and syntax of repmat() in Matlab it generates a large matrix consisting of an N-by-M tiling copies of A

Usage

repmat(A, N, M)

Arguments

A original matrix to be used as copies
N the number of rows of tiling copies of A
M the number of columns of tiling copies of A

Value

matrix consisting of an N-by-M tiling copies of A
Semi-Supervised Local Fisher Discriminant Analysis (SELF) for Semi-Supervised Dimensionality Reduction

Description

Performs semi-supervised local fisher discriminant analysis (SELF) on the given data. SELF is a linear semi-supervised dimensionality reduction method smoothly bridges supervised LFDA and unsupervised principal component analysis, by which a natural regularization effect can be obtained when only a small number of labeled samples are available.

Usage

```
self(X, Y, beta = 0.5, r, metric = c("orthonormalized", "plain", "weighted"), knn = 5, minObsPerLabel = 5)
```

Arguments

- **X**: n x d matrix of original samples. n is the number of samples.
- **Y**: length n vector of class labels
- **beta**: degree of semi-supervisedness (0 <= beta <= 1; default is 0.5) 0: totally supervised (discard all unlabeled samples) 1: totally unsupervised (discard all label information)
- **r**: dimensionality of reduced space (default: d)
- **metric**: type of metric in the embedding space (no default) 'weighted' — weighted eigenvectors 'orthonormalized' — orthonormalized 'plain' — raw eigenvectors
- **knn**: parameter used in local scaling method (default: 5)
- **minObsPerLabel**: the minimum number observations required for each different label (default: 5)

Value

- list of the SELF results:
  - **T**: d x r transformation matrix (Z = x * T)
  - **Z**: n x r matrix of dimensionality reduced samples

Author(s)

Yuan Tang
**References**


**See Also**

See 1fda for LFDA and klfda for the kernelized variant of LFDA (Kernel LFDA).

**Examples**

```r
## Not run:
x <- iris[, -5]
y <- iris[, 5]
result <- self(x, y, beta = 0.1, r = 3, metric = "plain")
## End(Not run)
```

---

**Negative One Half Matrix Power Operator**

**Description**

This function defines operation for negative one half matrix power operator.

**Usage**

```r
x %^% n
```

**Arguments**

- `x`: the matrix we want to operate on
- `n`: the exponent

**Value**

the matrix after negative one half power
## Index

+ Topic **discriminant**
  - klfda, 3
  - kmatrixGauss, 5
  - lfda, 6
  - self, 10
+ Topic **fisher**
  - klfda, 3
  - kmatrixGauss, 5
  - lfda, 6
  - self, 10
+ Topic **kernel**
  - kmatrixGauss, 5
+ Topic **klfda**
  - klfda, 3
  - kmatrixGauss, 5
+ Topic **lfda**
  - lfda, 6
  - self, 10
+ Topic **local**
  - klfda, 3
  - kmatrixGauss, 5
  - lfda, 6
  - self, 10
+ Topic **mahalanobis**
  - klfda, 3
  - kmatrixGauss, 5
  - lfda, 6
  - self, 10
+ Topic **metric**
  - klfda, 3
  - kmatrixGauss, 5
  - lfda, 6
  - self, 10
+ Topic **semi-supervised**
  - self, 10
+ Topic **transformation**
  - klfda, 3
  - kmatrixGauss, 5
  - lfda, 6
  - self, 10
  - %, 11
  - Cols, 2
  - getAffinityMatrix, 2
  - getMetricOfType, 3
  - klfda, 3, 7, 8, 11
  - kmatrixGauss, 3, 5
  - lfda, 3, 4, 6, 8, 11
  - plot.lfda, 7
  - predict.lfda, 8
  - print.lfda, 9
  - repmat, 9
  - self, 10