Package ‘rNMF’

February 20, 2015

Type Package

Title Robust Nonnegative Matrix Factorization

Description
An implementation of robust nonnegative matrix factorization (rNMF). The rNMF algorithm decomposes a nonnegative high dimension data matrix into the product of two low rank nonnegative matrices, while detecting and trimming outliers. The main function is rnmf(). The package also includes a visualization tool, see(), that arranges and prints vectorized images.

Version 0.5.0

Author Yifan Ethan Xu <`ethan.yifanxu@gmail.com`>, Jiayang Sun <`jsun@case.edu`>

Maintainer Yifan Ethan Xu <`ethan.yifanxu@gmail.com`>

Depends R (>= 2.14.0)

Imports nnls, knitr

VignetteBuilder knitr

License GPL (>= 2)

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2015-02-11 07:47:35

R topics documented:

face — *An Image of a Corrupted Face Image.*

Description

A 192 by 168 data matrix storing gray-scale pixel intensities of a corrupted face image. The original image was taken from Yale Face Database B [Georghiades et al. 2001] and then manually corrupted. Run the following line to see the image: image(t(face[ncol(face):1,]), axes = FALSE, col = grey(seq(0, 1, length = 256)), asp = 1)

Format

A data matrix containing 192 rows and 168 columns.

rnmf — *Robust Penalized Nonnegative Matrix Factorization (rNMF).*

Description

rNMF performs robust penalized nonnegative matrix factorizaion on a nonnegative matrix X to obtain low dimensional nonnegative matrices W and H, such that X ~ WH, while detecting and trimming different types of outliers in X.

Usage

```
rnmf(X, k = 5, alpha = 0, beta = 0, maxit = 20, tol = 0.001,
  gamma = FALSE, ini.W = NULL, ini.zeta = NULL, my.seed = NULL,
  variation = "cell", quiet = FALSE, nreg = 1, showprogress = TRUE)
```

Arguments

`X`	a 'p by n' nonnegative numerical matrix to be decomposed into X ~ WH.
`k`	the integer dimension to which X is projected (equals the number of columns of W and the number of rows of H, and smaller than min(p,n)). Default = 5.
`alpha`	a nonnegative number that controls the magnitude of W. The larger the alpha is, the higher penalty on ||W|| is, or ||W|| is forced to be smaller. Default = 0, representing no penalty on the magnitude ||W||.
`beta`	a nonnegative number that controls the sparsity of H. Default= 0. The larger the beta is, the higher penalty on sum_i ||H_j|| is, where H_j is the j-th column of H. Default = 0, representing no penalty on the sparsity of H.
`maxit`	the maximum number of iterations. Default = 20. The algorithm is done as follows: 1) fits W given H, then trims outliers in X, i.e. ones with large residuals from X - WH, then refits W without the outliers [this step can be repeated 'nreg' times, currently nreg = 1], then 2) repeat as in 1) with the roles of W and H swapped, and then iterate through 1) and 2) until convergence. Default = 20, allowing a maximum of 20 pairs of 1) and 2).

`tol`	the convergence tolerance. We suggest it to be a positive number smaller than 0.01. Default = 0.001.
`gamma`	a trimming percentage in (0, 1) of X. Default = 0.05 will trim 5% of elements of X (measured by cell, row or column as specified in 'variation'). If trim=0, there is no trim; so rNMF then performs the regular NMF.
`ini.W`	the initialization of the left matrix W, which is a "p by k" nonnegative numeric matrix. Default = NULL directs the algorithm to randomly generate an ini.W.
`ini.zeta`	a "p by n" logical matrix of True or False, indicating the initial locations of the outliers. The number of "TRUE"s in ini.zeta must be less than or equal to m = the rounded integer of (gamma * p * n). Default = NULL, initializes the cells as TRUE with the m largest residuals after the first iteration. Required only for "cell" trimming option.
`my.seed`	the random seed for initialization of W. If left to be NULL(default) a random seed is used. Required only if ini.W is not NULL.
`variation`	a character string indicating which trimming variation is used. The options are: 'cell' (default), 'col', 'row' and 'smooth'.
`quiet`	default = FALSE indicating a report would be given at the end of execution. If quiet = TRUE, no report is provide at the end.
`nreg`	the number of inner loop iterations [see 'maxit' above] to find outliers in X, given either H or W. Default = 1, is currently only implemented option in the "cell" variation.
`showprogress`	default = TRUE, shows a progress bar during iterations. If showprogress = FALSE, no progress bar is shown.

Details

rNMF decomposes a nonnegative p by n data matrix X as X ~ WH and detect and trims outliers. Here W and H are p by k and k by n nonnegative matrices, respectively; and k <= min{p, n} is the dimension of the subspace to which X is projected. The objective function is

||X - WH||_gamma + alpha * ||W||_2^2 + beta * sum(|H_.j|)^2

where alpha controls the magnitude of W, and beta controls the sparsity of H. The algorithm iteratively updates W, H and the outlier set zeta with alternating conditional nonnegative least square fittings until convergence.

Four variations of trimming are included in the algorithm: "cell", "row", "column" and "smooth". Specifically, the "cell" variation trims individual cell-wise outliers while "row" and "column" variations trim entire row or column outliers. The fourth variation "smooth" fills the cells that are declared outliers in each iteration by the average of the surrounding cells.

Value

An object of class 'rnmf', which is a list of the following items:

- W: left matrix of the decomposition X ~ WH, columns of which (i.e. W) are basis vectors of the low dimension projection.
- H: right matrix of the decomposition X ~ WH, columns of which (i.e. W) are low dimensional encoding of the data.

- fit: the fitted matrix W %*% H.
- trimmed: a list of locations of trimmed cells in each iteration.
- niter: the number of iterations performed.

Examples

```
## Load a clean single simulated tumor image.
data("tumor")
image(tumor) # look at the tumor image
dim(tumor) # it is a '70 by 70' matrix
## Add 5\% corruptions.
tumor.corrupted <- tumor
set.seed(1)
tumor.corrupted[sample(1:4900, round(0.05 * 4900), replace = FALSE)] <- 1
## Run rnmf with different settings
# No trimming
res.rnmf1 <- rnmf(X = tumor.corrupted, gamma = FALSE, my.seed = 1)
# 6 percent trimming, low dimension k = 5 (default)
res.rnmf2 <- rnmf(X = tumor.corrupted, tol = 0.001, gamma = 0.06, my.seed = 1)
# add sparsity constraint of H (beta = 0.1) with k = 10, and the "smooth" variation.
res.rnmf3 <- rnmf(X = tumor.corrupted, k = 10, beta = 0.1,
                  tol = 0.001, gamma = 0.06, my.seed = 1,
                  variation = "smooth", maxit = 30)

## Show results:
par(mfrow = c(2,2), mar = c(2,2,2,2))
image(tumor.corrupted, main = "Corrupted tumor image", xaxt = "n", yaxt = "n")
image(res.rnmf1$fit, main = "rnmf (no trimming) fit", xaxt = "n", yaxt = "n")
image(res.rnmf2$fit, main = "rnmf (cell) fit 2", xaxt = "n", yaxt = "n")
image(res.rnmf3$fit, main = "rnmf (smooth) fit 3", xaxt = "n", yaxt = "n")
```

---

see *Visualize Vectorized Images*

Description

The function is a wrapper of image(). It arranges and prints multiple or single images.

Usage

```
see(X, title = "Image", col = "heat", input = "multi", layout = "auto",
  ...)
```

Arguments

X	a numeric matrix. X is either a matrix where each column contains the pixels of a vectorized image, or simply the pixel matrix of one single image. The type of X is indicated by the argument 'input'.
title	a charactor string. Title of the graph.

`col`	a character string. Defult = "heat". What color scheme to use? Currently allows: • "heat" for heat color • "br" for (blue-cyan-green-yellow-red) palette • "grey" for grey scale
`input`	a charactor string with default = "multi", specifying the type of images in X. Possible options are: • "multi" if X contains multiple vectorized square images. • "single" if X is the matrix of a single image.
`layout`	a vector of 2 possible integers or a charactor string "auto" (default). If layout = "auto", multiple images will be arranged in an approximatedly 9 by 16 ratio. If layout = c(a,b), then images will be arranged in a rows and b columns.
`...`	further arguments to pass to image().

Details

If the input is a matrix of vectorized images (input = "multi", default setting), that is, each column contains pixels of one vectorized image, then see() restores each column into a matrix and show all images in one frame. Current version assumes the images are squared images. If the input is a matrix of one image (input = "single"), see() shows this image. Different color palette can be selected by specify the "col" argument. Build-in color palette includes greyscale, blue-red and heat color.

Examples

```
## Load a build-in data set Symbols, a 5625 by 30 matrix containing 30 75x75
## images.
data(Symbols)
see(Symbols, title = "Sample images of four symbols")
```

`Symbols` *30 Images Containing 4 Different Symbols.*

Description

A 5625 by 30 matrix. Each column stores vectorized pixels of a 75 by 75 image. Every image contains one or more of four shapes: circle, square, triangle and cross. The pixel intensities are standarized to 0 to 1. Run see(Symbols) to visualize.

Format

A 5625 by 30 numeric matrix.

`Symbols_c` *30 Images Containing 4 Different Symbols with Corruptions.*

Description

A 5625 by 30 matrix. Each column stores vectorized pixels of a 75 by 75 image. Every image contains one or more of four shapes: circle, square, triangle and cross. The pixel intensities are standarized to 0 to 1. Box corruptions are randomly added. Run see(Symbols_c) to visualize.

Format

A 5625 by 30 numeric matrix.

`tumor` *An Image of 3 Simulated Tumors.*

Description

A 70 by 70 data matrix storing pixel intensities of three tumors that are normalized to 0 to 1.

Format

A 70 by 70 numeric matrix.

Index