Package ‘logisticPCA’

October 13, 2022

Type Package
Title Binary Dimensionality Reduction
Version 0.2
Date 2016-03-13
NeedsCompilation yes
ByteCompile yes
Author Andrew J. Landgraf
Maintainer Andrew J. Landgraf <andland@gmail.com>
Description Dimensionality reduction techniques for binary data including logistic PCA.
License MIT + file LICENSE
LazyData true
URL https://github.com/andland/logisticPCA
Imports ggplot2
Suggests rARPACK (>= 0.10-0), testthat (>= 0.11.0), knitr, rmarkdown
VignetteBuilder knitr
RoxygenNote 5.0.1
Repository CRAN
Date/Publication 2016-03-14 07:54:24

R topics documented:

  logisticPCA-package ........................................ 2
  convexLogisticPCA .......................................... 2
  cv.clpca ....................................................... 4
  cv.lpca ........................................................ 5
  cv.lsvd ........................................................ 6
  fitted.lpca ................................................... 7
  fitted.lsvd .................................................... 7
  house_votes84 ............................................... 8
convexLogisticPCA

Description

Dimensionality reduction techniques for binary data by extending Pearson’s PCA formulation to minimize Binomial deviance. The convex relaxation to projection matrices, the Fantope, is used.

Usage

convexLogisticPCA(x, k = 2, m = 4, quiet = TRUE, partial_decomp = FALSE, max_iters = 1000, conv_criteria = 1e-06, random_start = FALSE, start_H, mu, main_effects = TRUE, ss_factor = 4, weights, M)
convexLogisticPCA

Arguments

- **x**: matrix with all binary entries
- **k**: number of principal components to return
- **m**: value to approximate the saturated model
- **quiet**: logical; whether the calculation should give feedback
- **partial_decomp**: logical; if TRUE, the function uses the rARPACK package to quickly initialize H when ncol(x) is large and k is small
- **max_iters**: number of maximum iterations
- **conv_criteria**: convergence criteria. The difference between average deviance in successive iterations
- **random_start**: logical; whether to randomly initialize the parameters. If FALSE, function will use an eigen-decomposition as starting value
- **start_H**: starting value for the Fantope matrix
- **mu**: main effects vector. Only used if main_effects = TRUE
- **main_effects**: logical; whether to include main effects in the model
- **ss_factor**: step size multiplier. Amount by which to multiply the step size. Quadratic convergence rate can be proven for ss_factor = 1, but I have found higher values sometimes work better. The default is ss_factor = 4. If it is not converging, try ss_factor = 1.
- **weights**: an optional matrix of the same size as the x with non-negative weights
- **M**: deprecated. Use m instead

Value

An S3 object of class clpca which is a list with the following components:

- **mu**: the main effects
- **H**: a rank k Fantope matrix
- **U**: a ceiling(k)-dimensional orthonormal matrix with the loadings
- **PCs**: the principal component scores
- **m**: the parameter inputed
- **iters**: number of iterations required for convergence
- **loss_trace**: the trace of the average negative log likelihood using the Fantope matrix
- **proj_loss_trace**: the trace of the average negative log likelihood using the projection matrix
- **prop_deviance_expl**: the proportion of deviance explained by this model. If main_effects = TRUE, the null model is just the main effects, otherwise the null model estimates 0 for all natural parameters.

References

Examples

```r
# construct a low rank matrix in the logit scale
rows = 100
cols = 10
set.seed(1)
mat_logit = outer(rnorm(rows), rnorm(cols))

# generate a binary matrix
mat = (matrix(runif(rows * cols), rows, cols) <= inv.logit(mat(mat_logit))) * 1.0

# run convex logistic PCA on it
clpca = convexLogisticPCA(mat, k = 1, m = 4)
```

---

**cv.clpca**

*CV for convex logistic PCA*

Description

Run cross validation on dimension and \( m \) for convex logistic PCA

Usage

```r
cv.clpca(x, ks, ms = seq(2, 10, by = 2), folds = 5, quiet = TRUE, Ms, ...)
```

Arguments

- `x` matrix with all binary entries
- `ks` the different dimensions \( k \) to try
- `ms` the different approximations to the saturated model \( m \) to try
- `folds` if `folds` is a scalar, then it is the number of folds. If it is a vector, it should be the same length as the number of rows in `x`
- `quiet` logical; whether the function should display progress
- `Ms` deprecation. Use `ms` instead
- `...` Additional arguments passed to `convexLogisticPCA`

Value

A matrix of the CV negative log likelihood with \( k \) in rows and \( m \) in columns

Examples

```r
# construct a low rank matrix in the logit scale
rows = 100
cols = 10
set.seed(1)
mat_logit = outer(rnorm(rows), rnorm(cols))
```
cv.lpca  

CV for logistic PCA

Description

Run cross validation on dimension and \(m\) for logistic PCA

Usage

cv.lpca(x, ks, ms = seq(2, 10, by = 2), folds = 5, quiet = TRUE, Ms, ...)

Arguments

- \(x\) matrix with all binary entries
- \(ks\) the different dimensions \(k\) to try
- \(ms\) the different approximations to the saturated model \(m\) to try
- \(folds\) if \(folds\) is a scalar, then it is the number of folds. If it is a vector, it should be the same length as the number of rows in \(x\)
- \(quiet\) logical; whether the function should display progress
- \(Ms\) deprecated. Use \(ms\) instead
- ... Additional arguments passed to logisticPCA

Value

A matrix of the CV negative log likelihood with \(k\) in rows and \(m\) in columns

Examples

# construct a low rank matrix in the logit scale
rows = 100
cols = 10
set.seed(1)
mat_logit = outer(rnorm(rows), rnorm(cols))

# generate a binary matrix
mat = (matrix(runif(rows * cols), rows, cols) <= inv.logit.mat(mat_logit)) * 1.0

## Not run:
negloglikes = cv.clpca(mat, ks = 1:9, ms = 3:6)
plot(negloglikes)

## End(Not run)
negloglikes = cv.lpca(mat, ks = 1:9, ms = 3:6)
plot(negloglikes)

## End(Not run)

---

### cv.lsvd

**CV for logistic SVD**

**Description**

Run cross validation on dimension for logistic SVD

**Usage**

```
cv.lsvd(x, ks, folds = 5, quiet = TRUE, ...)
```

**Arguments**

- `x` matrix with all binary entries
- `ks` the different dimensions k to try
- `folds` if folds is a scalar, then it is the number of folds. If it is a vector, it should be the same length as the number of rows in x
- `quiet` logical; whether the function should display progress
- `...` Additional arguments passed to logisticSVD

**Value**

A matrix of the CV negative log likelihood with k in rows

**Examples**

```r
# construct a low rank matrix in the logit scale
rows = 100
cols = 10
set.seed(1)
mat_logit = outer(rnorm(rows), rnorm(cols))

# generate a binary matrix
mat = (matrix(runif(rows * cols), rows, cols) <= inv.logit.mat(mat_logit)) * 1.0

## Not run:
negloglikes = cv.lsvd(mat, ks = 1:9)
plot(negloglikes)

## End(Not run)
```
fitted.lpca

### Description
Fit a lower dimensional representation of the binary matrix using logistic PCA

### Usage
```r
## S3 method for class 'lpca'
fitted(object, type = c("link", "response"), ...)
```

### Arguments
- `object`: logistic PCA object
- `type`: the type of fitting required. `type = "link"` gives output on the logit scale and `type = "response"` gives output on the probability scale
- `...`: Additional arguments

### Examples
```r
# construct a low rank matrix in the logit scale
rows = 100
cols = 10
set.seed(1)
mat_logit = outer(rnorm(rows), rnorm(cols))

# generate a binary matrix
mat = (matrix(runif(rows * cols), rows, cols) <= inv.logit.mat(mat_logit)) * 1.0

# run logistic PCA on it
lpca = logisticPCA(mat, k = 1, m = 4, main_effects = FALSE)

# construct fitted probability matrix
fit = fitted(lpca, type = "response")
```

---

fitted.lsvd

### Description
Fit a lower dimensional representation of the binary matrix using logistic SVD

### Usage
```r
## S3 method for class 'lsvd'
fitted(object, type = c("link", "response"), ...)
```
Arguments

object logistic SVD object
type the type of fitting required. type = "link" gives output on the logit scale and type = "response" gives output on the probability scale

Examples

# construct a low rank matrix in the logit scale
rows = 100
cols = 10
set.seed(1)
mat_logit = outer(rnorm(rows), rnorm(cols))

# generate a binary matrix
mat = (matrix(runif(rows * cols), rows, cols) <= inv.logit(mat(mat_logit))) * 1.0

# run logistic SVD on it
lsvd = logisticSVD(mat, k = 1, main_effects = FALSE, partial_decomp = FALSE)

# construct fitted probability matrix
fit = fitted(lsvd, type = "response")

---

house_votes84 United States Congressional Voting Records 1984

Description

This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition).

Usage

house_votes84

Format

A matrix with all binary or missing entries. There are 435 rows corresponding members of congress and 16 columns representing the bills being voted on. The row names refer to the political party of the members of congress.
inv.logit.mat

Source


Data converted to a matrix from:


Examples

data(house_votes84)
congress_lpca = logisticPCA(house_votes84, k = 2, m = 4)

______________________________
inv.logit.mat  Inverse logit for matrices
______________________________

Description

Apply the inverse logit function to a matrix, element-wise. It generalizes the inv.logit function from the gtools library to matrices

Usage

inv.logit.mat(x, min = 0, max = 1)

Arguments

x       matrix
min     Lower end of logit interval
max     Upper end of logit interval

Examples

(mat = matrix(rnorm(10 * 5), nrow = 10, ncol = 5))
inv.logit.mat(mat)
logisticPCA

Logistic Principal Component Analysis

Description

Dimensionality reduction for binary data by extending Pearson’s PCA formulation to minimize Binomial deviance

Usage

```r
logisticPCA(x, k = 2, m = 4, quiet = TRUE, partial_decomp = FALSE, 
max_iters = 1000, conv_criteria = 1e-05, random_start = FALSE, start_U, 
start_mu, main_effects = TRUE, validation, M, use_irlba)
```

Arguments

- `x`: matrix with all binary entries
- `k`: number of principal components to return
- `m`: value to approximate the saturated model. If \( m = 0 \), \( m \) is solved for
- `quiet`: logical; whether the calculation should give feedback
- `partial_decomp`: logical; if TRUE, the function uses the rARPACK package to more quickly calculate the eigen-decomposition. This is usually faster than standard eigen-decomposition when `ncol(x) > 100` and \( k \) is small
- `max_iters`: number of maximum iterations
- `conv_criteria`: convergence criteria. The difference between average deviance in successive iterations
- `random_start`: logical; whether to randomly initialize the parameters. If FALSE, function will use an eigen-decomposition as starting value
- `start_U`: starting value for the orthogonal matrix
- `start_mu`: starting value for mu. Only used if `main_effects = TRUE`
- `main_effects`: logical; whether to include main effects in the model
- `validation`: optional validation matrix. If supplied and \( m = 0 \), the validation data is used to solve for \( m \)
- `M`: deprecation. Use \( m \) instead
- `use_irlba`: deprecation. Use `partial_decomp` instead

Value

An S3 object of class `lpca` which is a list with the following components:

- `mu`: the main effects
- `U`: a \( k \)-dimensional orthonormal matrix with the loadings
- `PCs`: the principal component scores
logisticSVD

Description

Dimensionality reduction for binary data by extending SVD to minimize binomial deviance.

Usage

logisticSVD(x, k = 2, quiet = TRUE, max_iters = 1000, 
conv_criteria = 1e-05, random_start = FALSE, start_A, start_B, start_mu, 
partial_decomp = TRUE, main_effects = TRUE, use_irlba)
Arguments

- **x**: matrix with all binary entries
- **k**: rank of the SVD
- **quiet**: logical; whether the calculation should give feedback
- **max_iters**: number of maximum iterations
- **conv_criteria**: convergence criteria. The difference between average deviance in successive iterations
- **random_start**: logical; whether to randomly initialize the parameters. If FALSE, algorithm will use an SVD as starting value
- **start_A**: starting value for the left singular vectors
- **start_B**: starting value for the right singular vectors
- **start_mu**: starting value for mu. Only used if main_effects = TRUE
- **partial_decomp**: logical; if TRUE, the function uses the rARPACK package to more quickly calculate the SVD. When the number of columns is small, the approximation may be less accurate and slower
- **main_effects**: logical; whether to include main effects in the model
- **use_irlba**: deprecated. Use partial_decomp instead

Value

An S3 object of class lsvd which is a list with the following components:

- **mu**: the main effects
- **A**: a k-dimensional orthogonal matrix with the scaled left singular vectors
- **B**: a k-dimensional orthonormal matrix with the right singular vectors
- **iters**: number of iterations required for convergence
- **loss_trace**: the trace of the average negative log likelihood of the algorithm. Should be non-increasing
- **prop_deviance_expl**: the proportion of deviance explained by this model. If main_effects = TRUE, the null model is just the main effects, otherwise the null model estimates 0 for all natural parameters.

References


Examples

# construct a low rank matrix in the logit scale
rows = 100
cols = 10
set.seed(1)
mat_logit = outer(rnorm(rows), rnorm(cols))

# generate a binary matrix
mat = (matrix(runif(rows * cols), rows, cols) <= inv.logit(mat_logit)) * 1.0

# run logistic SVD on it
lsvd = logisticSVD(mat, k = 1, main_effects = FALSE, partial_decomp = FALSE)

# Logistic SVD likely does a better job finding latent features
# than standard SVD
plot(svd(mat_logit)$u[, 1], lsvd$A[, 1])
plot(svd(mat_logit)$u[, 1], svd(mat)$u[, 1])
Arguments

- **x**: convex logistic PCA object
- **type**: the type of plot (type = "trace" plots the algorithms progress by iteration, type = "loadings" plots the first 2 PC loadings, type = "scores" plots the first 2 PC scores)
- ... Additional arguments

Examples

```r
# construct a low rank matrix in the logit scale
rows = 100
cols = 10
set.seed(1)
mat_logit = outer(rnorm(rows), rnorm(cols))

# generate a binary matrix
mat = (matrix(runif(rows * cols), rows, cols) <= inv.logit(mat_logit)) * 1.0

# run convex logistic PCA on it
clpca = convexLogisticPCA(mat, k = 2, m = 4, main_effects = FALSE)

## Not run:
plot(clpca)
## End(Not run)
```

---

**plot.cv.lpca**

Plot CV for logistic PCA

**Description**

Plot cross validation results logistic PCA

**Usage**

```r
## S3 method for class 'cv.lpca'
plot(x, ...)
```

**Arguments**

- **x**: a cv.lpca object
- ... Additional arguments
Examples

# construct a low rank matrix in the logit scale
rows = 100
cols = 10
set.seed(1)
mat_logit = outer(rnorm(rows), rnorm(cols))

# generate a binary matrix
mat = (matrix(runif(rows * cols), rows, cols) <= inv.logit.mat(mat_logit)) * 1.0

## Not run:
negloglikes = cv.lpca(dat, ks = 1:9, ms = 3:6)
plot(negloglikes)

## End(Not run)

---

plot.lpca

Plot logistic PCA

Description

Plots the results of a logistic PCA

Usage

## S3 method for class 'lpca'
plot(x, type = c("trace", "loadings", "scores"), ...)

Arguments

x logistic PCA object
type the type of plot type = "trace" plots the algorithms progress by iteration, type = "loadings" plots the first 2 principal component loadings, type = "scores" plots the loadings first 2 principal component scores

... Additional arguments

Examples

# construct a low rank matrix in the logit scale
rows = 100
cols = 10
set.seed(1)
mat_logit = outer(rnorm(rows), rnorm(cols))

# generate a binary matrix
mat = (matrix(runif(rows * cols), rows, cols) <= inv.logit.mat(mat_logit)) * 1.0

# run logistic PCA on it
lpca = logisticPCA(mat, k = 2, m = 4, main_effects = FALSE)

## Not run:
plot(lpca)

## End(Not run)

---

**Plot logistic SVD**

**Description**

Plots the results of a logistic SVD

**Usage**

```r
## S3 method for class 'lsvd'
plot(x, type = c("trace", "loadings", "scores"), ...)
```

**Arguments**

- `x` logistic SVD object
- `type` the type of plot (default: `"trace"`)
  - `type = "trace"` plots the algorithms progress by iteration,
  - `type = "loadings"` plots the first 2 principal component loadings,
  - `type = "scores"` plots the loadings first 2 principal component scores
- `...` Additional arguments

**Examples**

```r
# construct a low rank matrix in the logit scale
rows = 100
cols = 10
set.seed(1)
mat_logit = outer(rnorm(rows), rnorm(cols))

# generate a binary matrix
mat = (matrix(runif(rows * cols), rows, cols) <= inv.logit(mat_logit)) * 1.0

# run logistic SVD on it
lsvd = logisticSVD(mat, k = 2, main_effects = FALSE, partial_decomp = FALSE)

## Not run:
plot(lsvd)

## End(Not run)
```
predict.clpca

Predict Convex Logistic PCA scores or reconstruction on new data

Description

Predict Convex Logistic PCA scores or reconstruction on new data

Usage

## S3 method for class 'clpca'
predict(object, newdata, type = c("PCs", "link", "response"), ...)  

Arguments

- **object**: convex logistic PCA object
- **newdata**: matrix with all binary entries. If missing, will use the data that object was fit on
- **type**: the type of fitting required. **type = "PCs"** gives the PC scores, **type = "link"** gives matrix on the logit scale and **type = "response"** gives matrix on the probability scale
- **...**: Additional arguments

Examples

```r
# construct a low rank matrices in the logit scale
rows = 100
cols = 10
set.seed(1)
loadings = rnorm(cols)
mat_logit = outer(rnorm(rows), loadings)
mat_logit_new = outer(rnorm(rows), loadings)

# convert to a binary matrix
mat = (matrix(runif(rows * cols), rows, cols) <= inv.logit.mat(mat_logit)) * 1.0
mat_new = (matrix(runif(rows * cols), rows, cols) <= inv.logit.mat(mat_logit_new)) * 1.0

# run logistic PCA on it
clpca = convexLogisticPCA(mat, k = 1, m = 4, main_effects = FALSE)

PCs = predict(clpca, mat_new)
```
predict.lpca  

**Predict Logistic PCA scores or reconstruction on new data**

**Description**

Predict Logistic PCA scores or reconstruction on new data

**Usage**

```r
## S3 method for class 'lpca'
predict(object, newdata, type = c("PCs", "link", "response"), ...)```

**Arguments**

- `object`: logistic PCA object
- `newdata`: matrix with all binary entries. If missing, will use the data that `object` was fit on
- `type`: the type of fitting required. `type = "PCs"` gives the PC scores, `type = "link"` gives matrix on the logit scale and `type = "response"` gives matrix on the probability scale
- `...`: Additional arguments

**Examples**

```r
# construct a low rank matrices in the logit scale
rows = 100
cols = 10
set.seed(1)
loadings = rnorm(cols)
mat_logit = outer(rnorm(rows), loadings)
mat_logit_new = outer(rnorm(rows), loadings)

# convert to a binary matrix
mat = (matrix(runif(rows * cols), rows, cols) <= inv.logit(mat_logit)) * 1.0
mat_new = (matrix(runif(rows * cols), rows, cols) <= inv.logit(mat_logit_new)) * 1.0

# run logistic PCA on it
lpca = logisticPCA(mat, k = 1, m = 4, main_effects = FALSE)

PCs = predict(lpca, mat_new)
```
Description

Predict Logistic SVD left singular values or reconstruction on new data

Usage

```r
## S3 method for class 'lsvd'
predict(object, newdata, quiet = TRUE, max_iters = 1000,
        conv_criteria = 1e-05, random_start = FALSE, start_A, type = c("PCs",
        "link", "response"), ...)
```

Arguments

- `object`: logistic SVD object
- `newdata`: matrix with all binary entries. If missing, will use the data that `object` was fit on
- `quiet`: logical; whether the calculation should give feedback
- `max_iters`: number of maximum iterations
- `conv_criteria`: convergence criteria. The difference between average deviance in successive iterations
- `random_start`: logical; whether to randomly initialize the parameters. If `FALSE`, algorithm implicitly starts $A$ with 0 matrix
- `start_A`: starting value for the left singular vectors
- `type`: the type of fitting required. `type = "PCs"` gives the left singular vectors, `type = "link"` gives matrix on the logit scale and `type = "response"` gives matrix on the probability scale
- `...`: Additional arguments

Details

Minimizes binomial deviance for new data by finding the optimal left singular vector matrix ($A$), given $B$ and $\mu$. Assumes the columns of the right singular vector matrix ($B$) are orthonormal.

Examples

```r
# construct a low rank matrices in the logit scale
rows = 100
cols = 10
set.seed(1)
loadings = rnorm(cols)
mat_logit = outer(rnorm(rows), loadings)
mat_logit_new = outer(rnorm(rows), loadings)
```
# convert to a binary matrix
mat = (matrix(runif(rows * cols), rows, cols) <= inv.logit.mat(mat_logit)) * 1.0
mat_new = (matrix(runif(rows * cols), rows, cols) <= inv.logit.mat(mat_logit_new)) * 1.0

# run logistic PCA on it
lsvd = logisticSVD(mat, k = 1, main_effects = FALSE, partial_decomp = FALSE)
A_new = predict(lsvd, mat_new)

---

**project.Fantope**  
*Project onto the Fantope*

**Description**  
Project a symmetric matrix onto the convex set of the rank k Fantope

**Usage**  
project.Fantope(x, k)

**Arguments**
- **x**: a symmetric matrix  
- **k**: the rank of the Fantope desired

**Value**
- **H**: a rank k Fantope matrix  
- **U**: a k-dimensional orthonormal matrix with the first k eigenvectors of H
Index

* datasets
  house_votes84, 8

* package
  logisticPCA-package, 2
  convexLogisticPCA, 2
  cv.clpca, 4
  cv.lpc, 5
  cv.lsvd, 6
  fitted.lpc, 7
  fitted.lsvd, 7
  house_votes84, 8
  inv.logit.mat, 9
  log_like_Bernoulli, 13
  logisticPCA, 10
  logisticPCA-package, 2
  logisticSVD, 11
  plot.clpca, 13
  plot.cv.lpc, 14
  plot.lpc, 15
  plot.lsvd, 16
  predict.clpca, 17
  predict.lpc, 18
  predict.lsvd, 19
  project.Fantope, 20