Package 'fastglmpca'

March 13, 2025

Encoding UTF-8

Type Package

Version 0.1-108

Date 2025-03-12

Title Fast Algorithms for Generalized Principal Component Analysis

Description Implements fast, scalable optimization algorithms for fitting generalized principal components analysis (GLM-PCA) models, as described in ``A Generalization of Principal Components Analysis to the Exponential Family" Collins M, Dasgupta S, Schapire RE (2002, ISBN:9780262271738), and subsequently ``Feature Selection and Dimension Reduction for Single-Cell RNA-Seq Based on a Multinomial Model" Townes FW, Hicks SC, Aryee MJ, Irizarry RA (2019) <doi:10.1186/s13059-019-1861-6>.

License GPL (>= 2)

URL https://github.com/stephenslab/fastglmpca

BugReports https://github.com/stephenslab/fastglmpca/issues

SystemRequirements GNU make

Depends R (>= 3.6)

Imports utils, Matrix, stats, distr, daarem, Rcpp (>= 1.0.8), RcppParallel (>= 5.1.5)

LinkingTo Rcpp, RcppArmadillo, RcppParallel

Suggests testthat, knitr, rmarkdown, ggplot2, cowplot

LazyData true

LazyDataCompression xz

NeedsCompilation yes

VignetteBuilder knitr

RoxygenNote 7.3.1

Author Eric Weine [aut, cre], Peter Carbonetto [aut], Matthew Stephens [aut]

Maintainer Eric Weine <ericweine15@gmail.com>

Repository CRAN

Date/Publication 2025-03-13 08:20:06 UTC

Contents

`fitted.glmpca_pois_fit`

Get Fitted Values for GLM-PCA Model Fit

Description

`fitted` method for the "glmpca_pois_fit" class.

Usage

```
## S3 method for class 'glmpca_pois_fit'
fitted(object, ...)
```

Arguments

`object`	An object of class "glmpca_fit", typically the result of calling `fit_glmpca_pois`.
`...`	Additional arguments passed to the generic `fitted` method.

Value

An n x p matrix of fitted means. Calculated as

$$exp(UDV')$$

using the `fit` object.

`fit_glmpca_pois` *Fit Poisson GLM-PCA Model to Count Data*

Description

Fit a Poisson GLM-PCA model by maximum-likelihood.

Usage

```
fit_glmpca_pois(
  Y,
  K,
  fit0 = init_glmpca_pois(Y, K),
  verbose = TRUE,
  control = list()
)

fit_glmpca_pois_control_default()

init_glmpca_pois(
  Y,
  K,
  U,
  V,
  X = numeric(0),
  Z = numeric(0),
  B = numeric(0),
  W = numeric(0),
  fixed_b_cols = numeric(0),
  fixed_w_cols = numeric(0),
  col_size_factor = TRUE,
  row_intercept = TRUE
)
```

Arguments

`Y`	The n x m matrix of counts; all entries of `Y` should be non-negative. It can be a sparse matrix (class `"dsparseMatrix"`) or dense matrix (class `"matrix"`).
`K`	Integer 1 or greater specifying the rank of the matrix factorization. This should only be provided if the initial fit (`fit0`) is not.
`fit0`	Initial model fit. It should be an object of class "glmpca_fit_pois", such as an output from `init_glmpca_pois` or a previous call to `fit_glmpca_pois`.
`verbose`	If `verbose = TRUE`, information about the algorithm's progress is printed after each update.
`control`	List of control parameters to modify behavior of the optimization algorithm; see "Details".

U	An optional argument giving the initial estimate of the loadings matrix. It should be an n x K matrix, where n is the number of rows in the counts matrix Y, and K > 0 is the rank of the matrix factorization. When U and V are not provided, input argument K should be specified instead.
V	An optional argument giving is the initial estimate of the factors matrix. It should be a m x K matrix, where m is the number of columns in the counts matrix Y, and K > 0 is the rank of the matrix factorization. When U and V are not provided, input argument K should be specified instead.
X	Optional argument giving row covariates of the count matrix Y. It should be an n x nx matrix, where nx is the number of row covariates.
Z	Optional argument giving column covariates of the count matrix Y. It should be an m x nz matrix, where nz is the number of column covariates.
B	Optional argument giving the initial estimates for the coefficients of the row covariates. It should be an m x nx matrix, where nx is the number of row covariates. This argument is ignored if X is not provided.
W	Optional argument giving the initial estimates for the coefficients of the column covariates. It should be an n x nz matrix, where nz is the number of column covariates. This argument is ignored if Z is not provided.
fixed_b_cols	Optional numeric vector specifying which columns of B (if any) should be fixed during optimization. This argument is ignored if X is not provided.
fixed_w_cols	Optional numeric vector specifying which columns of W (if any) should be fixed during optimization. This argument is ignored if Z is not provided.
col_size_factor	If col_size_factor = TRUE, add a fixed factor accounting for average differences in Poisson rates across columns of Y. Setting col_size_factor = TRUE and row_intercept = TRUE is intended to replicate the default behavior of glmpca.
row_intercept	If row_intercept = TRUE, add a factor accounting for average differences in Poisson rates across rows of Y. Setting col_size_factor = TRUE and row_intercept = TRUE is intended to replicate the default behavior of glmpca.

Details

In generalized principal component analysis (GLM-PCA) based on a Poisson likelihood, the counts y_{ij} stored in an $n \times m$ matrix Y are modeled as

$$y_{ij} \sim Pois(\lambda_{ij}),$$

in which the logarithm of each rate parameter λ_{ij} is defined as a linear combination of rank-K matrices to be estimated from the data:

$$\log \lambda_{ij} = (UDV')_{ij},$$

where U and V are orthogonal matrices of dimension $n \times K$ and $m \times K$, respectively, and D is a diagonal $K \times K$ matrix in which the entries along its diagonal are positive and decreasing. K is a tuning parameter specifying the rank of the matrix factorization. This is the same as the low-rank matrix decomposition underlying PCA (that is, the singular value decomposition), but because we are not using a linear (Gaussian) model, this is called "generalized PCA" or "GLM PCA".

To allow for additional components that may be fixed, `fit_glmpca_pois` can also fit the more general model

$$\log \lambda_{ij} = (UDV' + XB' + WZ')_{ij},$$

in which X, Z are fixed matrices of dimension $n \times n_x$ and $m \times n_z$, respectively, and B, W are matrices of dimension $m \times n_x$ and $n \times n_z$ to be estimated from the data.

`fit_glmpca_pois` computes maximum-likelihood estimates (MLEs) of U, V, D, B and W satisfying the orthogonality constraints for U and V and the additional constraints on D that the entries are positive and decreasing. This is accomplished by iteratively fitting a series of Poisson GLMs, where each of these individual Poissons GLMs is fitted using a fast "cyclic co-ordinate descent" (CCD) algorithm.

The `control` argument is a list in which any of the following named components will override the default optimization algorithm settings (as they are defined by `fit_glmpca_pois_control_default`). Additional control arguments not listed here can be used to control the behaviour of `fpiter` or `daarem`; see the help accompanying these functions for details.

`use_daarem` If `use_daarem = TRUE`, the updates are accelerated using DAAREM; see `daarem` for details.

`tol` This is the value of the "tol" control argument for `fpiter` or `daarem` that determines when to stop the optimization. In brief, the optimization stops when the change in the estimates or in the log-likelihood between two successive updates is less than "tol".

`maxiter` This is the value of the "maxiter" control argument for `fpiter` or `daarem`. In brief, it sets the upper limit on the number of CCD updates.

`convtype` This is the value of the "convtype" control argument for `daarem`. It determines whether the stopping criterion is based on the change in the estimates or the change in the log-likelihood between two successive updates.

`mon.tol` This is the value of the "mon.tol" control argument for `daarem`. This setting determines to what extent the monotonicity condition can be violated.

`training_frac` Fraction of the columns of input data `Y` to fit initial model on. If set to `1` (default), the model is fit by optimizing the parameters on the entire dataset. If set between `0` and `1`, the model is optimized by first fitting a model on a randomly selected fraction of the columns of `Y`, and then projecting the remaining columns of `Y` onto the solution. Setting this to a smaller value will increase speed but decrease accuracy.

`num_projection_ccd_iter` Number of co-ordinate descent updates be made to elements of `V` if and when a subset of `Y` is projected onto `U`. Only used if `training_frac` is less than `1`.

`num_ccd_iter` Number of co-ordinate descent updates to be made to parameters at each iteration of the algorithm.

`line_search` If `line_search = TRUE`, a backtracking line search is performed at each iteration of CCD to guarantee improvement in the objective (the log-likelihood).

`ls_alpha` alpha parameter for backtracking line search. (Should be a number between 0 and 0.5, typically a number near zero.)

`ls_beta` beta parameter for backtracking line search controlling the rate at which the step size is decreased. (Should be a number between 0 and 0.5.)

`calc_deriv` If `calc_deriv = TRUE`, the maximum gradient of U and V is calculated and stored after each update. This may be useful for assessing convergence of the optimization, though increases overhead.

`calc_max_diff` If `calc_max_diff = TRUE`, the largest change in U and V after each update is calculated and stored. This may be useful for monitoring progress of the optimization algorithm.

`orthonormalize` If `orthonormalize = TRUE`, the matrices U and V are made to be orthogonal after each update step. This improves the speed of convergence without the DAAREM acceleration; however, should not be used when `use_daarem = TRUE`.

You may use function `set_fastglmpca_threads` to adjust the number of threads used in performing the updates.

Value

An object capturing the state of the model fit. It contains estimates of U, V and D (stored as matrices `U`, `V` and a vector of diagonal entries `d`, analogous to the `svd` return value); the other parameters (X, B, Z, W); the log-likelihood achieved (`loglik`); information about which columns of B and W are fixed (`fixed_b_cols`, `fixed_w_cols`); and a data frame `progress` storing information about the algorithm's progress after each update.

References

Townes, F. W., Hicks, S. C., Aryee, M. J. and Irizarry, R. A. (2019). Feature selection and dimension reduction for single-cell RNA-Seq based on a multinomial model. *Genome Biology* **20**, 295. doi:10.1186/s1305901918616

Collins, M., Dasgupta, S. and Schapire, R. E. (2002). A generalization of principal components analysis to the exponential family. In *Advances in Neural Information Processing Systems* 14.

See Also

`fit_glmpca_pois`

Examples

```
set.seed(1)
n <- 200
p <- 100
K <- 3
dat  <- generate_glmpca_data_pois(n,p,K)
fit0 <- init_glmpca_pois(dat$Y,K)
fit  <- fit_glmpca_pois(dat$Y,fit0 = fit0)
```

---

generate_glmpca_data_pois

Generate Data from a GLM-PCA Model

Description

Generate data from a GLM-PCA model with a specified rank.

Usage

```
generate_glmpca_data_pois(n, p, K, link = c("log", "log1p"))
```

Arguments

n	Number of rows (genes).
p	Number of columns (cells).
K	Rank of the underlying mean structure.
link	Character vector describing the link between the product of the loading and factors and the mean of the data.

Details

This function assumes that each column of the data is generated from a multinomial distribution. Let

$$Y_j$$

denote column j of the generated data matrix. First, we set

$$sum(Y_j)$$

equal to a value generated from a

$$Uniform(50, 5000)$$

distribution. Then, we generate

$$L$$

and

$$F$$

from mixture distributions, and calculate

$$H = exp(L'F)$$

. Then, we generate the individual elements of

$$Y_j$$

from a multinomial model where the probability for each individual element is just

$$H_j$$

normalized.

Value

list with the following components

- LL - loadings of underlying mean structure. A K x n matrix
- FF - factors of underlying mean structure. A K x p matrix
- Y - n x p matrix of generated data.

Examples

```
set.seed(1)
sim_data <- generate_glmpca_data_pois(1000, 500, 1)
```

---

pbmc_facs *Mixture of 10 FACS-purified PBMC Single-Cell RNA-seq data*

Description

These data are a selection of the reference transcriptome profiles generated via single-cell RNA sequencing (RNA-seq) of 10 bead-enriched subpopulations of PBMCs (Donor A), described in Zheng *et al* (2017). The data are unique molecular identifier (UMI) counts for 16,791 genes in 3,774 cells. (Genes with no expression in any of the cells were removed.) Since the majority of the UMI counts are zero, they are efficiently stored as a 16,791 x 3774 sparse matrix. These data are used in the vignette illustrating how 'fastglmpca' can be used to analyze single-cell RNA-seq data. Data for a separate set of 1,000 cells is provided as a "test set" to evaluate out-of-sample predictions.

Format

`pbmc_facs` is a list with the following elements:

counts 16,791 x 3,774 sparse matrix of UMI counts, with rows corresponding to genes and columns corresponding to cells (samples). It is an object of class `"dgCMatrix"`).

counts_test UMI counts for an additional test set of 100 cells.

samples Data frame containing information about the samples, including cell barcode and source FACS population ("celltype" and "facs_subpop").

samples_test Sample information for the additional test set of 100 cells.

genes Data frame containing information and the genes, including gene symbol and Ensembl identifier.

fit GLM-PCA model that was fit to the UMI count data in the vignette.

Source

https://www.10xgenomics.com/resources/datasets

References

G. X. Y. Zheng *et al* (2017). Massively parallel digital transcriptional profiling of single cells. *Nature Communications* **8**, 14049. doi:10.1038/ncomms14049

Examples

```
library(Matrix)
data(pbmc_facs)
cat(sprintf("Number of genes: %d\n",nrow(pbmc_facs$counts)))
cat(sprintf("Number of cells: %d\n",ncol(pbmc_facs$counts)))
cat(sprintf("Proportion of counts that are non-zero: %0.1f%%.\n",
            100*mean(pbmc_facs$counts > 0)))
```

---

set_fastglmpca_threads

Set up Multithreading for fastglmpca

Description

Initialize RcppParallel multithreading using a pre-specified number of threads, or using the default number of threads when n is not specified or is NA.

Usage

```
set_fastglmpca_threads(n)
```

Arguments

n	The requested number of threads.

Value

The number of threads to be used.

---

summary.glmpca_pois_fit

Summarize GLM-PCA Model Fit

Description

summary method for objects of class "glmpcan_fit_pois".

Usage

```
## S3 method for class 'glmpca_pois_fit'
summary(object, ...)

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

Arguments

object	An object of class "glmpca_fit", typically the result of calling fit_glmpca_pois.
...	Additional arguments passed to the generic summary or print.summary method.
x	An object of class "summary.glmpca_fit", usually the result of a call to summary.glmpca_fit.

Value

summary returns a vector of basic statistics summarizing the model fit.

Index