Package ‘fastadi’

February 11, 2022

Type Package

Title Self-Tuning Data Adaptive Matrix Imputation

Version 0.1.0

Description Implements the AdaptiveImpute matrix completion algorithm of 'Intelligent Initialization and Adaptive Thresholding for Iterative Matrix Completion', <https://amstat.tandfonline.com/doi/abs/10.1080/10618600.2018.1518238>. AdaptiveImpute is useful for embedding sparsely observed matrices, often out performs competing matrix completion algorithms, and self-tunes its hyperparameter, making usage easy.

License MIT + file LICENSE

URL https://github.com/RoheLab/fastadi

BugReports https://github.com/RoheLab/fastadi/issues

Depends LRMF3, Matrix, R (>= 3.1)

Imports ellipsis, glue, logger, methods, Rcpp, RSpectra,

Suggests invertiforms, covr, knitr, rmarkdown, testthat

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

RoxygenNote 7.1.2

Config/testthat/edition 3

NeedsCompilation yes

Author Alex Hayes [aut, cre, cph] (<https://orcid.org/0000-0002-4985-5160>), Juhee Cho [aut], Donggyu Kim [aut], Karl Rohe [aut]

Maintainer Alex Hayes <alexpghayes@gmail.com>

Repository CRAN

Date/Publication 2022-02-11 08:20:02 UTC

R topics documented:

`adaptive_imputation` *Create an Adaptive Imputation object*

Description

`adaptive_imputation` objects are a subclass of `LRMF3::svd_like()`, with an additional field `alpha`.

Usage

```
adaptive_imputation(u, d, v, alpha, ...)
```

Arguments

`u`	A *matrix* "left singular-ish" vectors.
`d`	A *vector* of "singular-ish" values.
`v`	A *matrix* of "right singular-ish" vectors.
`alpha`	Value of `alpha` after final iteration.
`...`	Optional additional items to pass to the constructor.

Value

An `adaptive_imputation` object.

`adaptive_impute` *AdaptiveImpute*

Description

An implementation of the `AdaptiveImpute` algorithm for matrix completion for sparse matrices.

Usage

```
adaptive_impute(
  X,
  rank,
  ...,
  initialization = c("svd", "adaptive-initialize", "approximate"),
  max_iter = 200L,
  check_interval = 1L,
  epsilon = 1e-07,
  additional = NULL
)

## S3 method for class 'sparseMatrix'
adaptive_impute(
  X,
  rank,
  ...,
  initialization = c("svd", "adaptive-initialize", "approximate"),
  additional = NULL
)

## S3 method for class 'LRMF'
adaptive_impute(
  X,
  rank,
  ...,
  epsilon = 1e-07,
  max_iter = 200L,
  check_interval = 1L
)
```

Arguments

`X`	A sparse matrix of `Matrix::sparseMatrix()` class.
`rank`	Desired rank (integer) to use in the low rank approximation. Must be at least `2L` and at most the rank of `X`. Note that the rank of `X` is typically unobserved and computations may be unstable or even fail when `rank` is near or exceeds this threshold.
`...`	Unused additional arguments.
`initialization`	How to initialize the low rank approximation. Options are: • `"svd"` (default). In the initialization step, this treats unobserved values as zeroes. • `"adaptive-initialize"`. In the initialization step, this treats unobserved values as actually unobserved. However, the current `AdaptiveInitialize` implementation relies on dense matrix computations that are only suitable for relatively small matrices.

- "approximate". An approximate variant of AdaptiveInitialize that is less computationally expensive. See adaptive_initialize for details.

  Note that initialization matters as AdaptiveImpute optimizes a non-convex objective. The current theory shows that initializing with AdaptiveInitialize leads to a consistent estimator, but it isn't know if this is the case for SVD initialization. Empirically we have found that SVD initialization works well nonetheless.

max_iter — Maximum number of iterations to perform (integer). Defaults to 200L. In practice 10 or so iterations will get you a decent approximation to use in exploratory analysis, and and 50-100 will get you most of the way to convergence. Must be at least 1L.

check_interval — Integer specifying how often to perform convergence checks. Defaults to 1L. In practice, check for convergence requires a norm calculation that is expensive for large matrices and decreasing the frequency of convergence checks will reduce computation time. Can also be set to NULL, which case max_iter iterations of the algorithm will occur with no possibility of stopping due to small relative change in the imputed matrix. In this case delta will be reported as Inf.

epsilon — Convergence criteria, measured in terms of relative change in Frobenius norm of the full imputed matrix. Defaults to 1e-7.

additional — Ignored except when alpha_method = "approximate" in which case it controls the precise of the approximation to alpha. The approximate computation of alpha will always understand alpha, but the approximation will be better for larger values of additional. We recommend making additional as large as computationally tolerable.

Value

A low rank matrix factorization represented by an adaptive_imputation() object.

References

1. Cho, Juhee, Donggyu Kim, and Karl Rohe. "Asymptotic Theory for Estimating the Singular Vectors and Values of a Partially-Observed Low Rank Matrix with Noise." Statistica Sinica, 2018. https://doi.org/10.5705/ss.202016.0205.

2. ———. "Intelligent Initialization and Adaptive Thresholding for Iterative Matrix Completion: Some Statistical and Algorithmic Theory for Adaptive-Impute." Journal of Computational and Graphical Statistics 28, no. 2 (April 3, 2019): 323–33. https://doi.org/10.1080/10618600.2018.1518238.

Examples

```
mf <- adaptive_impute(ml100k, rank = 3L, max_iter = 5L)
mf
```

adaptive_initialize *AdaptiveInitialize*

Description

An implementation of the AdaptiveInitialize algorithm for matrix imputation for sparse matrices. At the moment the implementation is only suitable for small matrices with on the order of thousands of rows and columns at most.

Usage

```
adaptive_initialize(
  X,
  rank,
  ...,
  p_hat = NULL,
  alpha_method = c("exact", "approximate"),
  additional = NULL
)

## S3 method for class 'sparseMatrix'
adaptive_initialize(
  X,
  rank,
  ...,
  p_hat = NULL,
  alpha_method = c("exact", "approximate"),
  additional = NULL
)
```

Arguments

X	A sparse matrix of sparseMatrix class. Explicit (observed) zeroes in X can be dropped for
rank	Desired rank (integer) to use in the low rank approximation. Must be at least 2L and at most the rank of X.
...	Ignored.
p_hat	The portion of X that is observed. Defaults to NULL, in which case p_hat is set to the number of observed elements of X. Primarily for internal use in citation_impute() or advanced users.
alpha_method	Either "exact" or "approximate", defaulting to "exact". "exact" is computationally expensive and requires taking a complete SVD of matrix of size nrow(X) x nrow(X), and matches the AdaptiveInitialize algorithm exactly. "approximate" departs from the AdaptiveInitialization algorithm to compute a truncated SVD of rank rank + additional instead of a complete SVD. This reduces computational burden, but the resulting estimates of singular-ish values will not be penalized as much as in the AdaptiveInitialize algorithm.

additional — Ignored except when `alpha_method = "approximate"` in which case it controls the precise of the approximation to `alpha`. The approximate computation of `alpha` will always understand `alpha`, but the approximation will be better for larger values of `additional`. We recommend making `additional` as large as computationally tolerable.

Value

A low rank matrix factorization represented by an `adaptive_imputation()` object.

Examples

```
mf <- adaptive_initialize(
  ml100k,
  rank = 3,
  alpha_method = "approximate",
  additional = 2
)

mf
```

`citation_impute` *CitationImpute*

Description

An implementation of the `AdaptiveImpute` algorithm using efficient sparse matrix computations, specialized for the case when missing values in the upper triangle are taken to be *explicitly observed* zeros, as opposed to missing values. This is primarily useful for spectral decompositions of adjacency matrices of graphs with (near) tree structure, such as citation networks.

Usage

```
citation_impute(
  X,
  rank,
  ...,
  initialization = c("svd", "adaptive-initialize", "approximate"),
  max_iter = 200L,
  check_interval = 1L,
  epsilon = 1e-07,
  additional = NULL
)

## S3 method for class 'sparseMatrix'
citation_impute(
```

```
  X,
  rank,
  ...,
  initialization = c("svd", "adaptive-initialize", "approximate"),
  additional = NULL
)

## S3 method for class 'LRMF'
citation_impute(
  X,
  rank,
  ...,
  epsilon = 1e-07,
  max_iter = 200L,
  check_interval = 1L
)
```

Arguments

`X` A *square* sparse matrix of `Matrix::sparseMatrix()` class. Implicit zeros in the upper triangle of this matrix are considered observed and predictions on these elements contribute to the objective function minimized by `AdaptiveImpute`.

`rank` Desired rank (integer) to use in the low rank approximation. Must be at least `2L` and at most the rank of `X`. Note that the rank of `X` is typically unobserved and computations may be unstable or even fail when `rank` is near or exceeds this threshold.

`...` Unused additional arguments.

`initialization` How to initialize the low rank approximation. Options are:

- `"svd"` (default). In the initialization step, this treats unobserved values as zeroes.
- `"adaptive-initialize"`. In the initialization step, this treats unobserved values as actually unobserved. However, the current `AdaptiveInitialize` implementation relies on dense matrix computations that are only suitable for relatively small matrices.
- `"approximate"`. An approximate variant of `AdaptiveInitialize` that is less computationally expensive. See `adaptive_initialize` for details.

Note that initialization matters as `AdaptiveImpute` optimizes a non-convex objective. The current theory shows that initializing with `AdaptiveInitialize` leads to a consistent estimator, but it isn't know if this is the case for SVD initialization. Empirically we have found that SVD initialization works well nonetheless.

`max_iter` Maximum number of iterations to perform (integer). Defaults to `200L`. In practice 10 or so iterations will get you a decent approximation to use in exploratory analysis, and and 50-100 will get you most of the way to convergence. Must be at least `1L`.

`check_interval` Integer specifying how often to perform convergence checks. Defaults to `1L`. In practice, check for convergence requires a norm calculation that is expensive for large matrices and decreasing the frequency of convergence checks will reduce computation time. Can also be set to NULL, which case max_iter iterations of the algorithm will occur with no possibility of stopping due to small relative change in the imputed matrix. In this case delta will be reported as Inf.

epsilon
: Convergence criteria, measured in terms of relative change in Frobenius norm of the full imputed matrix. Defaults to 1e-7.

additional
: Ignored except when alpha_method = "approximate" in which case it controls the precise of the approximation to alpha. The approximate computation of alpha will always understand alpha, but the approximation will be better for larger values of additional. We recommend making additional as large as computationally tolerable.

Details

If OpenMP is available, citation_impute will automatically use getOption("Ncpus",1L) OpenMP threads to parallelize some key computations. Note that some computations are performed with the Armadillo C++ linear algebra library and may also be parallelized dependent on your BLAS and LAPACK installations and configurations.

Value

A low rank matrix factorization represented by an adaptive_imputation() object.

Examples

```
# create a (binary) square sparse matrix to demonstrate on

set.seed(887)

n <- 10
A <- rsparsematrix(n, n, 0.1, rand.x = NULL)

mf <- citation_impute(A, rank = 3L, max_iter = 1L, check_interval = Inf)
mf
```

Index