fbpca Documentation

Release 1.0

Facebook Inc

December 09, 2014

Contents

Functions for principal component analysis (PCA) and accuracy checks

---

This module contains eight functions:

pca principal component analysis (singular value decomposition)

eigens eigendecomposition of a self-adjoint matrix

eigenn eigendecomposition of a nonnegative-definite self-adjoint matrix

diffsnorm spectral-norm accuracy of a singular value decomposition

diffsnormc spectral-norm accuracy of a centered singular value decomposition

diffsnorms spectral-norm accuracy of a Schur decomposition

mult default matrix multiplication

set_matrix_mult re-definition of the matrix multiplication function "mult"

---

fbpca.**diffsnorm**(*A*, *U*, *s*, *Va*, *n_iter=20*)

2-norm accuracy of an approx to a matrix.

Computes an estimate snorm of the spectral norm (the operator norm induced by the Euclidean vector norm) of A - U diag(s) Va, using n_iter iterations of the power method started with a random vector; n_iter must be a positive integer.

Increasing n_iter improves the accuracy of the estimate snorm of the spectral norm of A - U diag(s) Va.

Parameters **A** : array_like

first matrix in A - U diag(s) Va whose spectral norm is being estimated

U : array_like

second matrix in A - U diag(s) Va whose spectral norm is being estimated

s : array_like

vector in A - U diag(s) Va whose spectral norm is being estimated

Va : array_like

fourth matrix in A - U diag(s) Va whose spectral norm is being estimated

n_iter : int, optional

number of iterations of the power method to conduct; n_iter must be a positive integer, and defaults to 20

Returns float

an estimate of the spectral norm of A - U diag(s) Va (the estimate fails to be accurate with exponentially low prob. as n_iter increases; see references DM1, DM2, and DM3 below)

See also:

diffsnormc, pca

Notes

To obtain repeatable results, reset the seed for the pseudorandom number generator.

References

[DM1], [DM2], [DM3]

Examples

```
>>> from fbpca import diffsnorm, pca
>>> from numpy.random import uniform
>>> from scipy.linalg import svd
>>>
>>> A = uniform(low=-1.0, high=1.0, size=(100, 2))
>>> A = A.dot(uniform(low=-1.0, high=1.0, size=(2, 100)))
>>> (U, s, Va) = svd(A, full_matrices=False)
>>> A = A / s[0]
>>>
>>> (U, s, Va) = pca(A, 2, True)
```

```
>>> err = diffsnorm(A, U, s, Va)
>>> print(err)
```

This example produces a rank-2 approximation U diag(s) Va to A such that the columns of U are orthonormal, as are the rows of Va, and the entries of s are all nonnegative and are nonincreasing. diffsnorm(A, U, s, Va) outputs an estimate of the spectral norm of A - U diag(s) Va, which should be close to the machine precision.

fbpca.**diffsnormc**(*A*, *U*, *s*, *Va*, *n_iter=20*)

2-norm approx error to a matrix upon centering.

Computes an estimate snorm of the spectral norm (the operator norm induced by the Euclidean vector norm) of C(A) - U diag(s) Va, using n_iter iterations of the power method started with a random vector, where C(A) refers to A from the input, after centering its columns; n_iter must be a positive integer.

Increasing n_iter improves the accuracy of the estimate snorm of the spectral norm of C(A) - U diag(s) Va, where C(A) refers to A after centering its columns.

Parameters **A** : array_like

first matrix in the column-centered A - U diag(s) Va whose spectral norm is being estimated

U : array_like

second matrix in the column-centered A - U diag(s) Va whose spectral norm is being estimated

s : array_like

vector in the column-centered A - U diag(s) Va whose spectral norm is being estimated

Va : array_like

fourth matrix in the column-centered A - U diag(s) Va whose spectral norm is being estimated

n_iter : int, optional

number of iterations of the power method to conduct; n_iter must be a positive integer, and defaults to 20

Returns float

an estimate of the spectral norm of the column-centered A - U diag(s) Va (the estimate fails to be accurate with exponentially low probability as n_iter increases; see references DC1, DC2, and DC3 below)

See also:

diffsnorm, pca

Notes

To obtain repeatable results, reset the seed for the pseudorandom number generator.

References

[DC1], [DC2], [DC3]

Examples

```
>>> from fbpca import diffsnormc, pca
>>> from numpy.random import uniform
>>> from scipy.linalg import svd
>>>
>>> A = uniform(low=-1.0, high=1.0, size=(100, 2))
>>> A = A.dot(uniform(low=-1.0, high=1.0, size=(2, 100)))
>>> (U, s, Va) = svd(A, full_matrices=False)
>>> A = A / s[0]
>>>
>>> (U, s, Va) = pca(A, 2, False)
>>> err = diffsnormc(A, U, s, Va)
>>> print(err)
```

This example produces a rank-2 approximation U diag(s) Va to the column-centered A such that the columns of U are orthonormal, as are the rows of Va, and the entries of s are nonnegative and nonincreasing. diffsnormc(A, U, s, Va) outputs an estimate of the spectral norm of the column-centered A - U diag(s) Va, which should be close to the machine precision.

fbpca.**diffsnorms**(*A*, *S*, *V*, *n_iter=20*)

2-norm accuracy of a Schur decomp. of a matrix.

Computes an estimate snorm of the spectral norm (the operator norm induced by the Euclidean vector norm) of A-VSV', using n_iter iterations of the power method started with a random vector; n_iter must be a positive integer.

Increasing n_iter improves the accuracy of the estimate snorm of the spectral norm of A-VSV'.

Parameters **A** : array_like

first matrix in A-VSV' whose spectral norm is being estimated

S : array_like

third matrix in A-VSV' whose spectral norm is being estimated

V : array_like

second matrix in A-VSV' whose spectral norm is being estimated

n_iter : int, optional

number of iterations of the power method to conduct; n_iter must be a positive integer, and defaults to 20

Returns float

an estimate of the spectral norm of A-VSV' (the estimate fails to be accurate with exponentially low probability as n_iter increases; see references DS1, DS2, and DS3 below)

See also:

eigenn, eigens

Notes

To obtain repeatable results, reset the seed for the pseudorandom number generator.

References

[DS1], [DS2], [DS3]

Examples

```
>>> from fbpca import diffsnorms, eigenn
>>> from numpy import diag
>>> from numpy.random import uniform
>>> from scipy.linalg import svd
>>>
>>> A = uniform(low=-1.0, high=1.0, size=(2, 100))
>>> A = A.T.dot(A)
>>> (U, s, Va) = svd(A, full_matrices=False)
>>> A = A / s[0]
>>>
>>> (d, V) = eigenn(A, 2)
>>> err = diffsnorms(A, diag(d), V)
>>> print(err)
```

This example produces a rank-2 approximation V diag(d) V' to A such that the columns of V are orthonormal and the entries of d are nonnegative and are nonincreasing. diffsnorms(A, diag(d), V) outputs an estimate of the spectral norm of A - V diag(d) V', which should be close to the machine precision.

fbpca.**eigenn**(*A*, *k=6*, *n_iter=4*, *l=None*)

Eigendecomposition of a NONNEGATIVE-DEFINITE matrix.

Constructs a nearly optimal rank-k approximation V diag(d) V' to a NONNEGATIVE-DEFINITE matrix A, using n_iter normalized power iterations, with block size l, started with an n x l random matrix, when A is n x n; the reference EGN below explains "nearly optimal." k must be a positive integer <= the dimension n of A, n_iter must be a nonnegative integer, and l must be a positive integer >= k.

The rank-k approximation V diag(d) V' comes in the form of an eigendecomposition – the columns of V are orthonormal and d is a real vector such that its entries are nonnegative and nonincreasing. V is n x k and len(d) = k, when A is n x n.

Increasing n_iter or l improves the accuracy of the approximation V diag(d) V'; the reference EGN below describes how the accuracy depends on n_iter and l. Please note that even n_iter=1 guarantees superb accuracy, whether or not there is any gap in the singular values of the matrix A being approximated, at least when measuring accuracy as the spectral norm || A - V diag(d) V' || of the matrix A - V diag(d) V' (relative to the spectral norm ||A|| of A).

Parameters **A** : array_like, shape (n, n)

matrix being approximated

k : int, optional

rank of the approximation being constructed; k must be a positive integer <= the dimension of A, and defaults to 6

n_iter : int, optional

number of normalized power iterations to conduct; n_iter must be a nonnegative integer, and defaults to 4

l : int, optional

block size of the normalized power iterations; l must be a positive integer >= k, and defaults to k+2

Returns **d** : ndarray, shape (k,)

vector of length k in the rank-k approximation V diag(d) V' to A, such that its entries are nonnegative and nonincreasing

V : ndarray, shape (n, k)

n x k matrix in the rank-k approximation V diag(d) V' to A, where A is n x n

See also:

diffsnorms, eigens, pca

Notes

THE MATRIX A MUST BE SELF-ADJOINT AND NONNEGATIVE DEFINITE.

To obtain repeatable results, reset the seed for the pseudorandom number generator.

The user may ascertain the accuracy of the approximation V diag(d) V' to A by invoking diffsnorms(A, numpy.diag(d), V).

References

[EGN]

Examples

```
>>> from fbpca import diffsnorms, eigenn
>>> from numpy import diag
>>> from numpy.random import uniform
>>> from scipy.linalg import svd
>>>
>>> A = uniform(low=-1.0, high=1.0, size=(2, 100))
>>> A = A.T.dot(A)
>>> (U, s, Va) = svd(A, full_matrices=False)
>>> A = A / s[0]
>>>
>>> (d, V) = eigenn(A, 2)
>>> err = diffsnorms(A, diag(d), V)
>>> print(err)
```

This example produces a rank-2 approximation V diag(d) V' to A such that the columns of V are orthonormal and the entries of d are nonnegative and nonincreasing. diffsnorms(A, diag(d), V) outputs an estimate of the spectral norm of A - V diag(d) V', which should be close to the machine precision.

fbpca.**eigens**(*A*, *k=6*, *n_iter=4*, *l=None*)

Eigendecomposition of a SELF-ADJOINT matrix.

Constructs a nearly optimal rank-k approximation V diag(d) V' to a SELF-ADJOINT matrix A, using n_iter normalized power iterations, with block size l, started with an n x l random matrix, when A is n x n; the reference EGS below explains "nearly optimal." k must be a positive integer <= the dimension n of A, n_iter must be a nonnegative integer, and l must be a positive integer >= k.

The rank-k approximation V diag(d) V' comes in the form of an eigendecomposition – the columns of V are orthonormal and d is a vector whose entries are real-valued and their absolute values are nonincreasing. V is n x k and len(d) = k, when A is n x n.

Increasing n_iter or l improves the accuracy of the approximation V diag(d) V'; the reference EGS below describes how the accuracy depends on n_iter and l. Please note that even n_iter=1 guarantees superb accuracy, whether or not there is any gap in the singular values of the matrix A being approximated, at least when measuring accuracy as the spectral norm || A - V diag(d) V' || of the matrix A - V diag(d) V' (relative to the spectral norm ||A|| of A).

Parameters **A** : array_like, shape (n, n)

matrix being approximated

k : int, optional

rank of the approximation being constructed; k must be a positive integer <= the dimension of A, and defaults to 6

n_iter : int, optional

number of normalized power iterations to conduct; n_iter must be a nonnegative integer, and defaults to 4

l : int, optional

block size of the normalized power iterations; l must be a positive integer >= k, and defaults to k+2

Returns **d** : ndarray, shape (k,)

vector of length k in the rank-k approximation V diag(d) V' to A, such that its entries are real-valued and their absolute values are nonincreasing

V : ndarray, shape (n, k)

n x k matrix in the rank-k approximation V diag(d) V' to A, where A is n x n

See also:

`diffsnorms`, `eigenn`, `pca`

Notes

THE MATRIX A MUST BE SELF-ADJOINT.

To obtain repeatable results, reset the seed for the pseudorandom number generator.

The user may ascertain the accuracy of the approximation V diag(d) V' to A by invoking diffsnorms(A, numpy.diag(d), V).

References

[EGS]

Examples

```
>>> from fbpca import diffsnorms, eigens
>>> from numpy import diag
>>> from numpy.random import uniform
>>> from scipy.linalg import svd
>>>
>>> A = uniform(low=-1.0, high=1.0, size=(2, 100))
```

```
>>> A = A.T.dot(A)
>>> (U, s, Va) = svd(A, full_matrices=False)
>>> A = A / s[0]
>>>
>>> (d, V) = eigens(A, 2)
>>> err = diffsnorms(A, diag(d), V)
>>> print(err)
```

This example produces a rank-2 approximation V diag(d) V' to A such that the columns of V are orthonormal, and the entries of d are real-valued and their absolute values are nonincreasing. diffsnorms(A, diag(d), V) outputs an estimate of the spectral norm of A - V diag(d) V', which should be close to the machine precision.

fbpca.**mult**(*A*, *B*)

default matrix multiplication.

Multiplies A and B together via the "dot" method.

Parameters **A** : array_like

first matrix in the product A*B being calculated

B : array_like

second matrix in the product A*B being calculated

Returns array_like

product of the inputs A and B

Examples

```
>>> from fbpca import mult
>>> from numpy import array
>>> from numpy.linalg import norm
>>>
>>> A = array([[1., 2.], [3., 4.]])
>>> B = array([[5., 6.], [7., 8.]])
>>> norm(mult(A, B) - A.dot(B))
```

This example multiplies two matrices two ways – once with mult, and once with the usual "dot" method – and then calculates the (Frobenius) norm of the difference (which should be near 0).

fbpca.**pca**(*A*, *k=6*, *raw=False*, *n_iter=2*, *l=None*)

Principal component analysis.

Constructs a nearly optimal rank-k approximation U diag(s) Va to A, centering the columns of A first when raw is False, using n_iter normalized power iterations, with block size l, started with a min(m,n) x l random matrix, when A is m x n; the reference PCA below explains "nearly optimal." k must be a positive integer <= the smaller dimension of A, n_iter must be a nonnegative integer, and l must be a positive integer >= k.

The rank-k approximation U diag(s) Va comes in the form of a singular value decomposition (SVD) – the columns of U are orthonormal, as are the rows of Va, and the entries of s are all nonnegative and nonincreasing. U is m x k, Va is k x n, and len(s)=k, when A is m x n.

Increasing n_iter or l improves the accuracy of the approximation U diag(s) Va; the reference PCA below describes how the accuracy depends on n_iter and l. Please note that even n_iter=1 guarantees superb accuracy, whether or not there is any gap in the singular values of the matrix A being approximated, at least when measuring accuracy as the spectral norm || A - U diag(s) Va || of the matrix A - U diag(s) Va (relative to the spectral norm ||A|| of A, and accounting for centering when raw is False).

Parameters **A** : array_like, shape (m, n)

matrix being approximated

k : int, optional

rank of the approximation being constructed; k must be a positive integer <= the smaller dimension of A, and defaults to 6

raw : bool, optional

centers A when raw is False but does not when raw is True; raw must be a Boolean and defaults to False

n_iter : int, optional

number of normalized power iterations to conduct; n_iter must be a nonnegative integer, and defaults to 2

l : int, optional

block size of the normalized power iterations; l must be a positive integer >= k, and defaults to k+2

Returns **U** : ndarray, shape (m, k)

m x k matrix in the rank-k approximation U diag(s) Va to A or C(A), where A is m x n, and C(A) refers to A after centering its columns; the columns of U are orthonormal

s : ndarray, shape (k,)

vector of length k in the rank-k approximation U diag(s) Va to A or C(A), where A is m x n, and C(A) refers to A after centering its columns; the entries of s are all nonnegative and nonincreasing

Va : ndarray, shape (k, n)

k x n matrix in the rank-k approximation U diag(s) Va to A or C(A), where A is m x n, and C(A) refers to A after centering its columns; the rows of Va are orthonormal

See also:

`diffsnorm`, `diffsnormc`, `eigens`, `eigenn`

Notes

To obtain repeatable results, reset the seed for the pseudorandom number generator.

The user may ascertain the accuracy of the approximation U diag(s) Va to A by invoking diffsnorm(A, U, s, Va), when raw is True. The user may ascertain the accuracy of the approximation U diag(s) Va to C(A), where C(A) refers to A after centering its columns, by invoking diffsnormc(A, U, s, Va), when raw is False.

References

[PCA]

Examples

```
>>> from fbpca import diffsnorm, pca
>>> from numpy.random import uniform
>>> from scipy.linalg import svd
>>>
>>> A = uniform(low=-1.0, high=1.0, size=(100, 2))
>>> A = A.dot(uniform(low=-1.0, high=1.0, size=(2, 100)))
>>> (U, s, Va) = svd(A, full_matrices=False)
>>> A = A / s[0]
>>>
>>> (U, s, Va) = pca(A, 2, True)
>>> err = diffsnorm(A, U, s, Va)
>>> print(err)
```

This example produces a rank-2 approximation U diag(s) Va to A such that the columns of U are orthonormal, as are the rows of Va, and the entries of s are all nonnegative and are nonincreasing. diffsnorm(A, U, s, Va) outputs an estimate of the spectral norm of A - U diag(s) Va, which should be close to the machine precision.

fbpca.**set_matrix_mult**(*newmult*)

re-definition of the matrix multiplication function "mult".

Sets the matrix multiplication function "mult" used in fbpca to be the input "newmult" – which must return the product A*B of its two inputs A and B, i.e., newmult(A, B) must be the product of A and B.

Parameters **newmult** : callable

matrix multiplication replacing mult in fbpca; newmult must return the product of its two array_like inputs

Returns None

Examples

```
>>> from fbpca import set_matrix_mult
>>>
>>> def newmult(A, B):
...     return A*B
...
>>> set_matrix_mult(newmult)
```

This example redefines the matrix multiplication used in fbpca to be the entrywise product.

[DM1] Jacek Kuczynski and Henryk Wozniakowski, Estimating the largest eigenvalues by the power and Lanczos methods with a random start, SIAM Journal on Matrix Analysis and Applications, 13 (4): 1094-1122, 1992.

[DM2] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert, Randomized algorithms for the low-rank approximation of matrices, Proceedings of the National Academy of Sciences (USA), 104 (51): 20167-20172, 2007. (See the appendix.)

[DM3] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert, A fast randomized algorithm for the approximation of matrices, Applied and Computational Harmonic Analysis, 25 (3): 335-366, 2008. (See Section 3.4.)

[DC1] Jacek Kuczynski and Henryk Wozniakowski, Estimating the largest eigenvalues by the power and Lanczos methods with a random start, SIAM Journal on Matrix Analysis and Applications, 13 (4): 1094-1122, 1992.

[DC2] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert, Randomized algorithms for the low-rank approximation of matrices, Proceedings of the National Academy of Sciences (USA), 104 (51): 20167-20172, 2007. (See the appendix.)

[DC3] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert, A fast randomized algorithm for the approximation of matrices, Applied and Computational Harmonic Analysis, 25 (3): 335-366, 2008. (See Section 3.4.)

[DS1] Jacek Kuczynski and Henryk Wozniakowski, Estimating the largest eigenvalues by the power and Lanczos methods with a random start, SIAM Journal on Matrix Analysis and Applications, 13 (4): 1094-1122, 1992.

[DS2] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert, Randomized algorithms for the low-rank approximation of matrices, Proceedings of the National Academy of Sciences (USA), 104 (51): 20167-20172, 2007. (See the appendix.)

[DS3] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert, A fast randomized algorithm for the approximation of matrices, Applied and Computational Harmonic Analysis, 25 (3): 335-366, 2008. (See Section 3.4.)

[EGN] Nathan Halko, Per-Gunnar Martinsson, and Joel Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, arXiv:0909.4061 [math.NA; math.PR], 2009 (available at arXiv).

[EGS] Nathan Halko, Per-Gunnar Martinsson, and Joel Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, arXiv:0909.4061 [math.NA; math.PR], 2009 (available at arXiv).

[PCA] Nathan Halko, Per-Gunnar Martinsson, and Joel Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, arXiv:0909.4061 [math.NA; math.PR], 2009 (available at arXiv).

f

D

E

F

M

P

S