Lecture 8, Sci. Comp. for DPhil Students

Nick Trefethen, Thursday 7.11.19

Today

- II.8 Matrix factorizations
- II.9 SVD and low-rank approximation
- II.10 Gaussian elimination as an iterative algorithm

Handouts

- Questionnaire
- `m16_factorizations.m` and `m17_svd.m`
- "Gaussian elimination as an iterative algorithm" (*SIAM News*, 2012)
- Assignment 3

Announcements

- Please fill in the questionnaire
- Read: Trefethen & Bau chapters 4-5

This is the last of our eight lectures on numerical linear algebra. I hope I have persuaded you that this material is the foundation for all kinds of scientific computing.

Today we're going to survey matrix factorisations at a high level and then turn to the singular value decomposition, or SVD.

II.8 Matrix factorizations

Most algorithms of dense numerical linear algebra (1) compute a matrix factorization, then (2) solve a resulting sequence of simpler problems (triangular, orthogonal, diagonal, tridiagonal,. . .). You could say this is the *central dogma of numerical linear algebra*:

```
algorithms <-> matrix factorizations
```

The standard methods for computing these factorizations do it by introducing zeros one by one until the desired structure is reached. They are all backward stable, which means: they give the exact factorization of a matrix $A + \Delta A$ with $\Delta A = O(10^{-16})$ times the size (the norm) of A.

Here are the seven most famous and important factorizations. We assume A is square. As usual we also assume A is real, though everything generalizes to the complex case. QR, LU, and SVD also generalize to A rectangular.

QR factorization
$A = QR$. Q orthogonal, R upper-triangular.
Used for least-squares and as step in iterative algs. for eigenvalues and SVD.

LU factorization
$PA = LU$. L unit lower-triang., U upper-triang., P a permutation matrix.
Result of Gaussian elimination with row pivoting.
Used for systems of eqs. and low-rank matrix approximation (II.10)

Cholesky factorization
$A = R^T R$. A SPD, R upper-triangular.
Used for SPD systems of equations.

Eigenvalue factorization
$A = VDV^{-1}$. A diagonalizable, V nonsingular, D diagonal.
Computed by **QR algorithm** ($\neq$ QR factorization).

Orthogonal eigenvalue factorization
$A = QDQ^T$. Q orthogonal, A symmetric Does not exist for most matrices.

Schur factorization
$A = QTQ^T$. Q orthogonal, T upper-triangular, A arbitrary.
Every matrix has a Schur factorization.
The eigenvalues of A are the diagonal entries of T.

Singular value decomposition (SVD)
$A = USV^T$. U and V orthogonal, S diagonal and ≥ 0, A arbitrary.
Every matrix has an SVD.

`[m16_factorizations.m]`

II.9 SVD and low-rank approximation

(Trefethen & Bau chapters 4-5)

Singular values are related to eigenvalues and equally important, but less well-known among pure mathematicians. The reason is that eigenvalues are a concept of algebra, invariant with respect to change of basis, whereas singular values are a concept of analysis, norm-dependent.

Eigenvalues are useful for problems involving powers or exponentials of A
(stability, resonance,...).

Singular values are useful for problems involving A or A^{-1} itself
(least-squares, conditioning, low-rank aproximation,...).

The (reduced) SVD for $m \times n$ A ($m \geq n$) is a factorization

$$A = U\Sigma V^T$$

where:

U is $m \times n$ with orthonormal columns,

Σ is $n \times n$ diagonal with decreasing entries ≥ 0,

V is $n \times n$ orthogonal.

Here is the basic fact that SVD encapsulates:

Every $m \times n$ matrix A maps the unit ball in R^n to a hyperellipsoid in R^m.

A **hyperellipsoid** is the higher-dimensional generalization of an ellipsoid.
It's an m-dimensional sphere, except stretched linearly by various factors in various directions.

Here's the figure that explains it all (here for $m = n = 2$):

[Hand-drawn. Shows a circle mapping to an ellipse. In the circle, two orthogonal axes are labeled v_1 and v_2. In the ellipse, the major and minor axes are labeled $\sigma_1 u_1$ and $\sigma_2 u_2$.]

Principal semiaxes of hyperellipsoid: $\sigma_1 u_1, \ldots, \sigma_n u_n$

singular values of A: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$

left singular vectors of A: $u_1, \ldots, u_n$ (orthonormal)

The preimages of $\{\sigma_j u_j\}$ are an orthonormal set $\{v_j\}$.

right singular vectors of A: $v_1, \ldots, v_n$ (orthonormal)

```
 --------------   -------------       -------------   --------------
|              | |   |     |   |     |   |     |   | |s             |
|              | |   |     |   |     |   |     |   | | 1            |
|              | |   |     |   |     |   |     |   | |   s          |
|              | |   |     |   |     |   |     |   | |    2         |
|              | | v | ... | v |  =  |   |     |   | |      .       |
|              | |  1|     |  n|     |   |     |   | |        .     |
|              | |   |     |   |     | u | ... | u | |          .   |
|      A       | |   |     |   |     |  1|     |  n| |            s |
|              | |   |     |   |     |   |     |   | |             n|
|              |  -------------      |   |     |   |  --------------
|              |                     |   |     |   |
|              |                     |   |     |   |
|              |                     |   |     |   |
|              |                     |   |     |   |
 --------------                       -------------
```

SVD as change of basis in square case $m = n$

$$AV = U\Sigma, \quad \text{i.e.,} \quad A = U\Sigma V^T.$$

If $b = Ax$, then $b = U\Sigma V^T x$, i.e.

```
    T                           T
  ( U b )        =      Sigma ( V x )

coeffs of b                 coeffs of x
in basis of                 in basis of
left singular               right singular
```

`vectors` `vectors`

Thus after distinct orthogonal changes of basis in both domain and range,
A becomes diagonal.

Some facts for general A:

- Every A has an SVD.
- Singular values are unique, but not singular vectors.
- $\|A\|_2 = \sigma_1$ (2-norm, which we haven't defined)
- $\|A\|_F = (\sum_j \sigma_j^2)^{1/2}$ (Frobenius norm — likewise)
- $\{$singular values of $A\} = \{$square roots of eigenvalues of $A^T A\}$

rank(A) = number of nonzero singular values

range(A) = span$\{u_1, \ldots, u_r\}$ where $r = \text{rank}(A)$

Some facts for square A:

- $\prod_j \sigma_j = \prod_j |\lambda_j| = |\det(A)|$
- $\|A^{-1}\|_2 = 1/\sigma_n$
- $\text{cond}(A) = \|A\|_2 \|A^{-1}\|_2 = \sigma_1/\sigma_n$.

Low-rank approximation

Let $r = \text{rank}(A) \leq n$.

Then from the SVD we easily verify

$$A = \sum_{j=1}^{r} \sigma_j u_j v_j^T$$

(i.e., the SVD exhibits A as a sum of rank-1 matrices).

For any $k < r$ define

$$A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^T.$$

A_k is a rank-k matrix.

Moreover, it is the *closest* rank-k matrix to A in both the 2-norm and the Frobenius norm.

This has applications all over the place.

It also has generalizations to infinite dimensional operators and matrices in functional analysis. We find "s-numbers" and "Schmidt pairs", and:

compact operator: one whose singular values decay to zero

Hilbert-Schmidt operator: one whose singular values have a finite sum-of-squares.

`[m17_svd.m]`

II.10 Gaussian elimination as an iterative algorithm

GE, the standard algorithm for solving $Ax = b$, is the archetypical *direct algorithm* of numerical linear algebra.

It has recently been noticed that GE is also an archetype of an *iterative algorithm* in data science: a fast algorithm for low-rank approximation or "poor man's SVD".

Usually GE is done with "partial pivoting" — row interchanges at each step. We'll speak however of the variant of "column pivoting" — row and column interchanges at each step. (This has a better guarantee of numerical stability, though not much different in practice, hence rarely used since it requires more work.)

Direct GE

A is $n \times n$, n not too big.

It must be nonsingular and hopefully not too ill-conditioned.

```
for k = 1:n
  Find largest entry, say a_{ij}
  Subtract off rank 1 matrix A(:,j)*A(i,:)/a_{ij}
end
```

Iterative GE

A is $m \times n$, m and/or n huge. It *must* be ill-conditioned for this to be useful.

Same algorithm! — but now, stop when the matrix that remains is sufficiently small.

In Chebfun: `f = cheb.gallery2('smokering'); explain(f)`