Numerical Linear Algebra, Part II

Massimo Ricotti

ricotti@astro.umd.edu

University of Maryland

LU decomposition

- Suppose we can write $\mathbf{A}$ as a product of two matrices: $\mathbf{A} = \mathbf{LU}$, where $\mathbf{L}$ is lower triangular and $\mathbf{U}$ is upper triangular:

$$\mathbf{L} = \begin{pmatrix} \times & 0 & 0 \\ \times & \times & 0 \\ \times & \times & \times \end{pmatrix} \qquad \mathbf{U} = \begin{pmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{pmatrix}$$

- Then $\mathbf{Ax} = (\mathbf{LU})\,\mathbf{x} = \mathbf{L}\,(\mathbf{Ux}) = \mathbf{b}$, i.e., must solve,
  (1) $\mathbf{Ly} = \mathbf{b}$; (2) $\mathbf{Ux} = \mathbf{y}$.
- Can reuse $\mathbf{L}$ and $\mathbf{U}$ for subsequent calculations.
- Why is this better?
  - Solving triangular matrices is easy: just use forward substitution for (1), backsubstitution for (2).

- Problem is, how to decompose $\mathbf{A}$ into $\mathbf{L}$ and $\mathbf{U}$?
  - Expand matrix multiplication $\mathbf{LU}$ to get n^2 equations for $n^2 + n$ unknowns (elements of $\mathbf{L}$ and $\mathbf{U}$ plus n extras because diagonal elements counted twice).
  - Get an extra n equations by choosing $L_{ii} = 1$ ($i = 1, n$).
  - Then use Crout's algorithm for finding solution to these $n^2 + n$ equations "trivially" (*NRiC* §2.3).

LU decomposition in *NRiC*

- The routines `ludcmp()` and `lubksb()` perform LU decomposition and backsubstituion, respectively.
- Can easily compute $\mathbf{A}^{-1}$ (solve for the identity matrix column by column) and $\det(\mathbf{A})$ (find the product of the diagonal elements of the LU decomposed matrix)—see *NRiC* §2.3.
- *Warning*: for large matrices, computing $\det(\mathbf{A})$ can overflow or underflow the computer's floating-point dynamic range (there are ways around this).

Iterative Improvement

- For large sets of linear equations $\mathbf{Ax} = \mathbf{b}$, roundoff error may become a problem.
- We want to know $\mathbf{x}$ but we only have $\mathbf{x} + \delta\mathbf{x}$, which is an exact solution to $\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b}$.
- Subtract the first equation from the second, and use the second to eliminate $\delta\mathbf{b}$:
$$\mathbf{A}\delta\mathbf{x} = \mathbf{A}(\mathbf{x} + \delta\mathbf{x}) - \mathbf{b}.$$
- The RHS is known, hence can solve for $\delta\mathbf{x}$. Subtract this from the wrong solution to get an improved solution (make sure to use `doubles`!). See `mprove()` in *NRiC*.

Tridiagonal matrices

- Many systems can be written as (or reduced to):

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i \qquad i = 1, n$$

  i.e., a tridiagonal matrix:

$$\begin{bmatrix} b_1 & c_1 & & & & 0's \\ a_2 & b_2 & c_2 & & & \\ & a_3 & b_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_{n-1} & b_{n-1} & c_{n-1} \\ 0's & & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}.$$

  Here a_1 and c_n are associated with “boundary conditions” (i.e., x_0 and x_{n+1}).

- LU decomposition and backsubstitution is very efficient for tri-di systems: $\mathcal{O}(n)$ operations as opposed to $\mathcal{O}(n^3)$ in general case.

Sparse matrices

- Operations on many sparse systems in general can be optimized, e.g.,
  tridiagonal;
  band diagonal with bandwidth M;
  block diagonal;
  banded.
- See *NRiC* §2.7 for various systems and techniques.

Iterative methods

- For very large systems, direct solution methods (e.g., LU decomposition) are slow and RE prone.
- Often iterative methods much more efficient:
  1. Guess a trial solution $\mathbf{x}^0$.
  2. Compute a correction $\mathbf{x}^1 = \mathbf{x}^0 + \delta\mathbf{x}$.
  3. Iterate procedure until convergence, i.e., $|\delta\mathbf{x}| < \Delta$.
- E.g., congugate gradient method for sparse systems (*NRiC* §2.7).

Singular value decomposition

- Can diagnose or (nearly) solve singular or near-singular systems.
- Used for solving linear least-squares problems.
- Theorem: any $m \times n$ matrix $\mathbf{A}$ (with m rows and n columns) can be written:

$$\mathbf{A} = \mathbf{U}\mathbf{W}\mathbf{V}^{\mathrm{T}},$$

  where $\mathbf{U}$ ($m \times n$) and $\mathbf{V}$ ($n \times n$) are orthogonal[a] and $\mathbf{W}$ ($n \times n$) is a diagonal matrix.
- Proof: buy a good linear algebra textbook ... or/and look on wikipedia:
  `http://en.wikipedia.org/wiki/Singular_value_decomposition`

---

[a] $\mathbf{U}$ has orthonormal columns while $\mathbf{V}$, being square, has both orthonormal rows and columns.

- The n diagonal values w_i of $\mathbf{W}$ are zero or positive and are called the "singular values."
- The *NRiC* routine `svdcmp()` returns $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ given $\mathbf{A}$. You have to trust it (or test it yourself!).
  - Uses Householder reduction, QR diagonalization, etc.
- If $\mathbf{A}$ is square, then we know[a]

$$\mathbf{A}^{-1} = \mathbf{V}\left[\mathrm{diag}(1/w_i)\right]\mathbf{U}^{\mathrm{T}}.$$

  - This is fine so long as no w_i is too small (or zero). Otherwise, the presence of small or zero w_i tell you how singular your system is...

[a] Since $\mathbf{U}$ and $\mathbf{V}$ are square and orthogonal, their inverses are equal to their transposes, and since $\mathbf{W}$ is diagonal, its inverse is a diagonal matrix whose elements are $1/w_i$.

Definitions

- Condition number $\mathrm{cond}(\mathbf{A}) = (\max w_i)/(\min w_i)$.
  - If $\mathrm{cond}(\mathbf{A}) = \infty$, $\mathbf{A}$ is singular.
  - If $\mathrm{cond}(\mathbf{A})$ very large ($\sim e_m^{-1}$), $\mathbf{A}$ is ill-conditioned.
- Consider $\mathbf{Ax} = \mathbf{b}$. If $\mathbf{A}$ is singular, there is some subspace of $\mathbf{x}$ (the nullspace) such that $\mathbf{Ax} = \mathbf{0}$.
- The nullity of $\mathbf{A}$ is the dimension of the nullspace (the number of linearly independent vectors $\mathbf{x}$ that can be found in it).
- The subspace of $\mathbf{b}$ such that $\mathbf{Ax} = \mathbf{b}$ is the range of $\mathbf{A}$.
- The rank of $\mathbf{A}$ is the dimension of the range.

The homogeneous equation

- SVD constructs orthonormal bases for the nullspace and range of a matrix.
- Columns of $\mathbf{U}$ with corresponding non-zero w_i are an orthonormal basis for the range.
- Columns of $\mathbf{V}$ with corresponding zero w_i are an orthonormal basis for the nullspace.
- Hence immediately have solution for $\mathbf{A}\mathbf{x} = \mathbf{0}$, i.e., the columns of $\mathbf{V}$ with corresponding zero w_i.

Residuals

- If $\mathbf{b}$ ($\neq \mathbf{0}$) lies in the range of $\mathbf{A}$, then the singular equations do in fact have a solution.
- Even if $\mathbf{b}$ is outside the range of $\mathbf{A}$, can get solution which minimizes <u>residual</u> $r = |\mathbf{A}\mathbf{x} - \mathbf{b}|$.
  - Trick: replace $1/w_i$ by 0 if $w_i = 0$ and compute

$$\mathbf{x} = \mathbf{V}\left[\mathrm{diag}(1/w_i)\right](\mathbf{U}^{\mathrm{T}}\mathbf{b}).$$

- Similarly, can set $1/w_i = 0$ if w_i very small.

Approximation of matrices

- Can write $\mathbf{A} = \mathbf{U}\mathbf{W}\mathbf{V}^{\mathrm{T}}$ as

$$A_{ij} = \Sigma_{k=1}^{N} w_k U_{ik} V_{jk}.$$

- If most of the singular values w_k are small, then $\mathbf{A}$ is well-approximated by only a few terms in the sum (strategy: sort w_k's in descending order).
- For large memory savings, just store the columns of $\mathbf{U}$ and $\mathbf{V}$ corresponding to non-negligible w_k's.
- Useful technique for digital image processing.

SVD examples

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

SVD:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Nullity = 2, nullspace = (1,0,0),(0,1,0), rank = 1, range = (0,0,1).

Inverse:

$$\mathbf{A}^{-1} = \begin{pmatrix} \infty & 0 & 0 \\ 0 & \infty & 0 \\ 0 & 0 & \frac{1}{4} \end{pmatrix}.$$

- (1,2,1) not in range of A. But, e.g., $\mathbf{b}$ = (0,0,8) is, with an infinite number of solutions $\mathbf{x}$, i.e., (0,0,2), (1,0,2), (1,1,2), etc.
- The solution that has the smallest magnitude $|\mathbf{x}|$ in this case is (0,0,2), applying the formula for minimizing the residual.
- The solution closest to $\mathbf{b}$ = (1,2,1) is (0,0,1/4) using the same technique.

Gray square in image processing

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} \sqrt{\frac{2}{3}} & 0 & \sqrt{\frac{1}{3}} \\ -\sqrt{\frac{1}{6}} & \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{3}} \\ -\sqrt{\frac{1}{6}} & -\sqrt{\frac{1}{2}} & \sqrt{\frac{1}{3}} \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} \sqrt{\frac{1}{6}} & \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{3}} \\ \sqrt{\frac{1}{6}} & -\sqrt{\frac{1}{2}} & \sqrt{\frac{1}{3}} \\ -\sqrt{\frac{2}{3}} & 0 & \sqrt{\frac{1}{3}} \end{pmatrix}$$

Just store rightmost columns of $\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$, and the one non-zero element of $\mathbf{W}$. In general, for a gray $N \times N$ image, need only store $2N + 1$ numbers, instead of N^2.