Lecture 11

Announcements

- Practice exam posted
- Review sessions

Week 6 : PCA + Least Sq.
7 : least sq. + exam
8 inference
9 prediction
10 visualization + manifold learning

Outline

- Solving systems of linear equations
  - LU decomposition
- Numerical linear algebra
  - FLOPs
  - condition number
- Least squares problems
  - SVD
  - Cholesky
  - QR
  - Regularization

Solving linear systems (Golub and Van Loan Ch. 3)

$$\begin{aligned}3x_1 + 5x_2 &= 9\\ 6x_1 + 7x_2 &= 4\end{aligned} \Rightarrow \begin{aligned}3x_1 + 5x_2 &= 9\\ -3x_2 &= -14\end{aligned}$$ (Gaussian elimination)

$$\begin{bmatrix}3 & 5\\ 6 & 7\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} = \begin{bmatrix}9\\ 4\end{bmatrix}$$

Decompose $\begin{bmatrix}3 & 5\\ 6 & 7\end{bmatrix} = \begin{bmatrix}1 & 0\\ 2 & 1\end{bmatrix}\begin{bmatrix}3 & 5\\ 0 & -3\end{bmatrix}$ (LU decomposition) $\sim 2n^3/3$ flops

Generally speaking, the LU decomposition is the best way to solve systems of linear equations $A\vec{x} = \vec{b}$ where $A \in \mathbb{R}^{n\times n}$ and has independent columns.

$A\vec{x} = \vec{b}$ becomes $A\vec{x} = LU\vec{x} = L\vec{y} = \vec{b}$

1. Solve $L\vec{y} = \vec{b}$
2. Solve $U\vec{x} = \vec{y}$

triangular systems

Forward substitution for lower triangular systems

ex $\begin{bmatrix}\ell_{11} & 0\\ \ell_{21} & \ell_{22}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} = \begin{bmatrix}b_1\\ b_2\end{bmatrix}$

$$\left.\begin{aligned}x_1 &= b_1/\ell_{11}\\ x_2 &= (b_2 - \ell_{21}x_1)/\ell_{22}\end{aligned}\right\}$$ ok as long as $\ell_{11}\ell_{22} \neq 0$

in general $x_i = \left(b_i - \sum_{j=1}^{i-1} \ell_{ij}x_j\right)/\ell_{ii}$ $\sim n^2$ flops

Backward substitution for upper triangular systems

go the other direction

$$x_i = \left(b_i - \sum_{j=i+1}^{n} u_{ij}x_j\right)/u_{ii}$$ $\sim n^2$ flops

Why bother? We could have just computed A^{-1}?

The standard approach to computing A^{-1} already involves computing LU

- inversion is not faster than LU
- inversion is often less accurate than LU.

Numerical linear algebra

Practically speaking, how do we develop efficient algorithms to answer matrix algebra question?

ex. What is the best/most accurate way to compute SVD?
or to solve $A\vec{x} = \vec{b}$?

Especially important with big matrices

Floating point number

Memory is limited so we cannot store numbers with infinite precision.

Floating point numbers consist of significands and bases

Double (FP64)

Sign bit (1 bit) + Exponent (11 bits) + Significand (52 bits)

FLOP (Floating point operation)

FLOPs is a measure of the complexity of a task.

ex Adding two length-n vector element wise : n flops

dot product : $\sim 2n$ flops

Solving $A\vec{x} = \vec{b}$ for diagonal A : n flops

Solving $A\vec{x} = \vec{b}$ for general A : $O(n^3)$ flops

Condition number

Sensitivity of square systems (Golub + Van Loan 2.6)

let $A\vec{x}=\vec{b}$ with $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$, A has lin. ind. columns

Using SVD, we can rewrite

$$A = \sum_{i=1}^{n} \sigma_i \vec{u}_i \vec{v}_i^T = U\Sigma V^T$$

Then $\vec{x} = A^{-1}\vec{b} = (U\Sigma V^T)^{-1}\vec{b} = \sum_{i=1}^{n} \frac{\vec{u}_i^T\vec{b}}{\sigma_i}\vec{v}_i$ (exercise)

If σ_n is small, then small changes in A or $\vec{b}$ will yield large changes in $\vec{x}$.

In fact σ_n is $\|\cdot\|_2$ distance from A to the set of singular matrices

As $\sigma_n \to 0$, $\vec{x}$ will be increasingly sensitive to perturbations.

Consider the problem

$$(A+\varepsilon F)\vec{x}(\varepsilon) = \vec{b} + \varepsilon\vec{f}$$

note; let $\vec{x}(0) = \vec{x}$

$F \in \mathbb{R}^{n\times n}$, $\vec{f} \in \mathbb{R}^n$

If A is nonsingular, $\vec{x}(\varepsilon)$ is differentiable in a neighborhood of 0

The vector of derivatives is

$$\dot{\vec{x}}(0) = A^{-1}(\vec{f} - F\vec{x})$$

(exercise $\to$ use the chain rule)

Using a Taylor approximation,

$$\vec{x}(\varepsilon) = \vec{x} + \varepsilon\dot{\vec{x}}(0) + O(\varepsilon^2)$$

$$\boxed{\frac{\|\vec{x}(\varepsilon)-\vec{x}\|}{\|\vec{x}\|} \le \varepsilon\|A^{-1}\|\left\{\frac{\|\vec{f}\|}{\|\vec{x}\|} + \|F\|\right\} + O(\varepsilon^2)}$$

relative error in $\vec{x}$

def For square matrices A, define the condition number $\kappa(A)$ to $\kappa(A) = \|A\| \|A^{-1}\|$ with $\kappa(A) = \infty$ for singular A.

So if $\rho_A = |\varepsilon| \dfrac{\|F\|}{\|A\|}$ and $\rho_{\vec{b}} = |\varepsilon| \dfrac{\|\vec{f}\|}{\|\vec{b}\|}$ are relative errors

$$\frac{\|\vec{x}(\varepsilon) - \vec{x}\|}{\|\vec{x}\|} \leq \kappa(A)(\rho_A + \rho_{\vec{b}}) + O(\varepsilon^2)$$ (exercise)

Thus $\kappa(A)$ quantifies the sensitivity of $A\vec{x} = \vec{b}$

note $\kappa(A)$ depends on the norm

$$\kappa_2(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\sigma_1}{\sigma_n} = \frac{\sigma_{max}(A)}{\sigma_{min}(A)}$$

note if $\kappa(A)$ is large, A is "ill-conditioned"

note If Q is orthogonal, what is $\kappa_2(Q)$? 1

What if there is no exact solution to $A\vec{x} = \vec{b}$?

Least squares choose $\hat{x}$ to minimize $\|\vec{b} - A\hat{x}\|_2^2$

Four approaches

- Pseudoinverse (SVD)
- Solving normal equations with Cholesky decomposition.
- Using $A = QR$ decomposition
- Minimizing $\|\vec{b} - A\hat{x}\|_2^2 + \delta^2 \|\vec{x}\|_2^2$ (add a penalty term)

Note. minimizing $\|\vec{b} - A\hat{x}\|_2^2$ is minimizing $(\vec{b} - A\hat{x})^T (\vec{b} - A\hat{x})$ is equivalent to solving the normal equations $A^T A \hat{x} = A^T \vec{b}$

Note $A^T A$ is symmetric but potentially large and ill-conditioned