Introduction of Matrix Computation

Topics covered

Computational problems:

① Solve square linear system

$$Ax = b,$$

where A is an $n \times n$ matrix and the unknown is an $n \times 1$ vector.

② Solve least-squares system

$$Ax \approx b,$$

where A is an $m \times n$ $(m \geq n)$ matrix and the unknown is $n \times 1$.

③ Eigenvalue and Eigenvector problem

$$Av = \lambda v,$$

where A is an $n \times n$ matrix, λ is a number and v is an $n \times 1$ vector.

④ Singular value and singular vector problem

$$\begin{cases} Av = \sigma u \\ A^T u = \sigma v, \end{cases}$$

where A is an $m \times n$ matrix, σ is a positive number, and u, v are $m \times 1$ and $n \times 1$ vectors.

Computational methods:

- Matrix factorizations: LU, QR, Singular Value Decomposition (SVD), ...
- Iterative Methods: Gauss-Seidel, Kaczmarz, Gradient Descent, Conjugate Graidient, ...
- Randomized Numerical Algorithms,

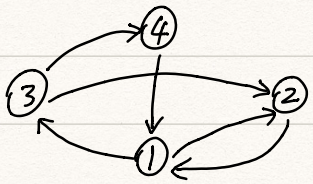
Applications:

- Image processing / Computer vision
- Data Analysis / Machine learning

Why these problems are important?

Example 1: Adjancency Matrix and Google's PageRank

Network is ubiquitous in our life, which can be modeled as a graph.



Rank the nodes? (Applications in Google PageRank / Ranking of teams in a league)

Cut the graph into small ones? (Application in Community Detection and Image Segmentation)

The answer of these problems are closely related to adjancency matrix of the graph. An adjancency matrix G is defined as:

$$g_{ij} = \begin{cases} 1 & \text{if node } j \text{ points to node } i, \\ 0 & \text{otherwise.} \end{cases}$$

The adjancency matrix for the graph above is

$$G = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The answer of the questions is related to solve a linear equation or eigenvalue problem involving a normalized version of G.

Example 2: Regression

Regression is one of the most important topics in statistics, which is about fitting a function to the sampled data in order to explore the relationship between two quantities. For example, we want to use a cubic polynomial to fit a few pairs of

We want to find a cubic polynomial to fit the given data

$$(x_i, y_i) \qquad i = 1, 2, \cdots, 8.$$

That is, we want to find a polynomial

$$y = c_0 + c_1 x + c_2 x^2 + c_3 x^3$$

such that

$$y_i \approx c_0 + c_1 x_i + c_2 x_i^2 + c_3 x_i^3, \qquad i = 1, 2, \cdots, 8.$$

In matrix notations, this problem is rewritten as

$$\begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_8 & x_8^2 & x_8^3 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} \approx \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_8 \end{pmatrix},$$

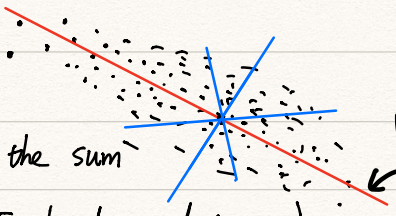
4-vector; 8-vector; 8×4 matrix

which is indeed a least-squares problem.

Example 3: Principal Component Analysis

Suppose we are given some data points, modeled by n $m \times 1$ vectors $\{x_i\}_{i=1}^n$ with x_i is an $m \times 1$ vector for $i = 1, \cdots, n$.

Which direction is the most important?



To answer this question, we project all data points onto different lines, and then we compute the sum of the projection errors (in terms of squared Eucleadian distance) of all data points.

Clearly, the direction of the line that minimizes the projection error is the most important direction. For example, the red line in the figure.

What are the next, the next next important directions? .....

It is solved by the singular value decomposition (SVD) of the matrix $X = [x_1\ x_2\ \cdots\ x_n]$. This is known as principal component analysis (PCA) in data analysis / machine learning.

Example 4: Subroutine of more complex algorithms

Consider the smooth unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x).$$

To solve this problem, let x_k be the current estimate, we do Taylor's expansion of $f(x)$ around x_k to get

$$f(x) \approx f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2} (x - x_k)^T \nabla^2 f(x_k) (x - x_k),$$

where $\nabla f(x_k)$ is the gradient and $\nabla^2 f(x_k)$ is the Hessian. Then, in Newton's method, instead of minimizing $f(x)$ directly, we minimize the right hand side of the Taylor's expansion. That is, we solve

$$\nabla f(x_k) + \nabla^2 f(x_k)(x - x_k) = 0,$$

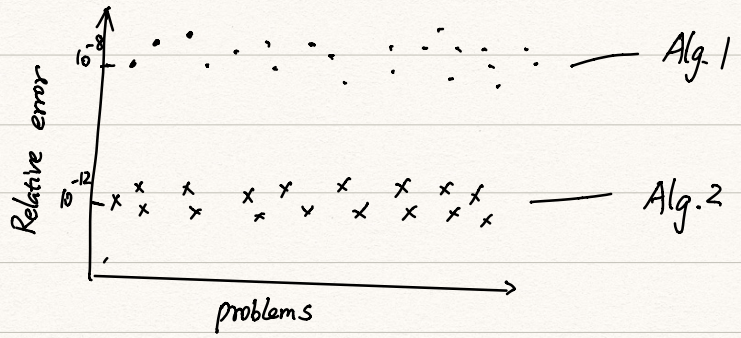
which is a linear equation.

Actually, solving linear equations plays a fundamental role in 2nd order optimization algorithms.

What are emphasized?

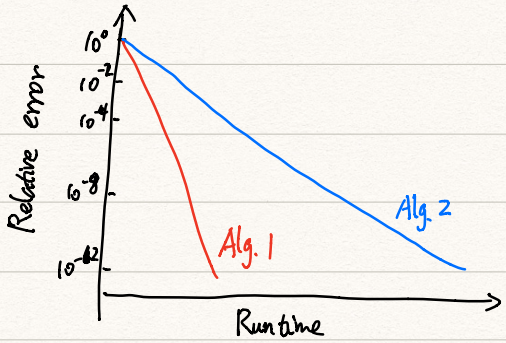
- Mathematical foundations for linear equations, least squares systems, and various matrix decompositions.
- How to solve matrix computation problems numerically? — Algorithms
  - Stability and accuracy

    Example 1: Which algorithm is more accurate?

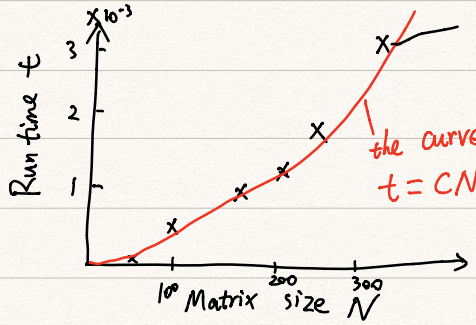


- Runtime

  Example II: Which algorithm uses less runtime for the same accuracy?



  Example III: Runtime Vs. Problem Size



  Runtime Vs. N for computing Ax, where A is $N \times N$ and x is $N \times 1$

  - Memory
  - ...

- Algorithms are the most important.
- Won't emphasize too much on the exact analysis of stability and accuracy but just give rule of thumb and use numerical results to demonstrate core ideas
- Solve the case-study problems using taught algorithms.