SI231b: Matrix Computations

Lecture 1: brief overview

邱越

qiuyue@shanghaitech.edu.cn

School of Information Science and Technology

ShanghaiTech University

Sept. 5, 2022

Course Information

General Information

- Instructor: Yue Qiu
  - Office: SIST 2-403
  - E-mail: qiuyue@shanghaitech.edu.cn
- Lecture hours and venue: Week 1 - 12,
  - Monday 15: 00 – 16:40, Teaching Center 101
  - Wednesday 15: 00 – 16:40, Teaching Center 101
- We use Piazza for Q&A, please join our Piazza under

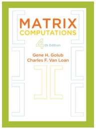
  https://piazza.com/shanghaitech.edu.cn/fall2022/si231b

Course Contents

- Foundation course on matrix analysis and computations, widely used in many different fields, e.g.,
  - machine learning, computer vision
  - systems and control, signal and image processing, communications, networks
  - optimization, and many more...
- **Aim:** topics on matrix analysis and computations at an advanced or research level.
- **Scope:**
  - basic matrix concepts, subspace, norms
  - linear system of equations, LU factorization, Cholesky factorization
  - linear least squares, QR decomposition, pseudo-inverse
  - eigenvalue decomposition (ED), singular value decomposition (SVD), symmetric positive definite (SPD) matrices, low-rank approximations......
  - iterative methods for linear systems......

Learning Resources

- Lecture slides will be uploaded to Blackboard and Piazza
- Textbook:
  - Gene H. Golub and Charles F. van Loan, *Matrix Computations* (Fourth Edition), Johns Hopkins University Press, 2013. (人民邮电出版社影印版 available)
  - Gene H. Golub and Charles F. van Loan 著，程晓亮译, 矩阵计算，人民邮电出版社，2020.



Learning Resources

- References:
  - if you prefer numerical computations
    - Lloyd N. Trefethen and David Bau III. *Numerical Linear Algebra*, Society for Industrial and Applied Mathematics (SIAM), 1997.
    - James Demmel. *Applied Numerical Linear Algebra*, SIAM, 1997.

      Math 221, UC Berkeley, Spring 2022.
  - or if you prefer analysis
    - Roger A. Horn and Charles R. Johnson, *Matrix Analysis* (Second Edition), Cambridge University Press, 2012.
    - Roger A. Horn and Charles R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, 1991.
  - or you need to revisit linear algebra
    - Gilbert Strang. *Linear Algebra and its Applications* (fourth edition), Cengage Learning, 2006.
    - Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*, SIAM, 2005.

Assessment

- Assignments: 30%, 5 times
  - may contain Matlab/Python programming tasks
  - where to submit: Gradescope
    - you should use LaTeX for typesetting
    - for better adjustment, you are allowed to use Microsoft Office for typesetting for the first two assignments
    - hand-written submission is not accepted
  - for flexibility, you are given six extra days in total for late submission
- Midterm exam: 40%
  - close book
  - only one page cheat sheet of A4 size is allowed
- Final project: 30%, working in a group of 2 or 1

Your final score = 0.3*assignments + 0.4*Midterm exam + 0.3*project

Undergraduates and Graduates share the same Assessment Criteria

Academic Honesty

- Academic honesty: you are strongly advised to read
  - 上海科技大学学生学术诚信规范与管理办法（试行）
  - 信息学院学术诚信问题的处罚细则（试行）

  You are assumed to understand the aspects described therein.
- In this course,
  - you are encouraged to discuss in groups for better understanding, but you must finish your assignments independently.
  - plagiarism is never allowed. You should protect your assignments solution at any time.
  - once similar solutions, codes showed in the submissions, you will be invited to go through suspected plagiarism investigation.

Teaching Assistants

图 1: 孟宇煌

图 2: 李斌

图 3: 黄建国

Additional Notice

- Sitting in is welcome, and please register on BB to keep you updated.
- Office hour (OH)
  - Yue Qiu: Wednesday 13:30 – 14:30, SIST 2-403
  - TAs:
    - To be announced
- Do check your (ShanghaiTech) Email regularly, this is the only way we can reach you.

A Glimpse of Topics

Problem: given $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{y} \in \mathbb{R}^n$, solve

$$\mathbf{A}\mathbf{x} = \mathbf{y}.$$

- **Question 1:** How to solve it?
  - no answers like
    - `x=inv(A)*y` or `x= A\y` in Matlab!
    - `x=np.dot(np.linalg.inv(A), y)`
  - this is about matrix computations
- **Question 2:** How to solve it when n is very large?
  - it's too slow to use `backslash x= A\y` when n is very large
  - getting better understanding of matrix computations will enable you to exploit problem structures to build efficient solvers
- **Question 3:** How sensitive is the solution $\mathbf{x}$ when $\mathbf{A}$ and $\mathbf{y}$ contain errors?
  - key to system analysis, or building robust solutions

Problem: given $\mathbf{A} \in \mathbb{R}^{m \times n}$ ($m > n$), $\mathbf{y} \in \mathbb{R}^m$, solve

$$\min_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2,$$

where $\|\cdot\|_2$ is the Euclidean norm; i.e., $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$.

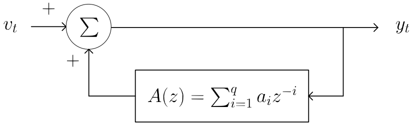
- widely used in science, engineering
- assuming a tall and full-rank $\mathbf{A}$, the LS solution is uniquely given by

$$\mathbf{x}_{\mathsf{LS}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}.$$

- let $\{y_t\}_{t\geq 0}$ be a time series.
- **Model** (autoregressive (AR) model):

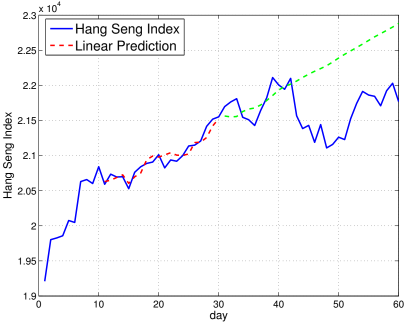
$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_q y_{t-q} + v_t, \quad t = 0, 1, 2, \ldots$$

for some coefficients $\{a_i\}_{i=1}^q$, where v_t is noise or modeling error.



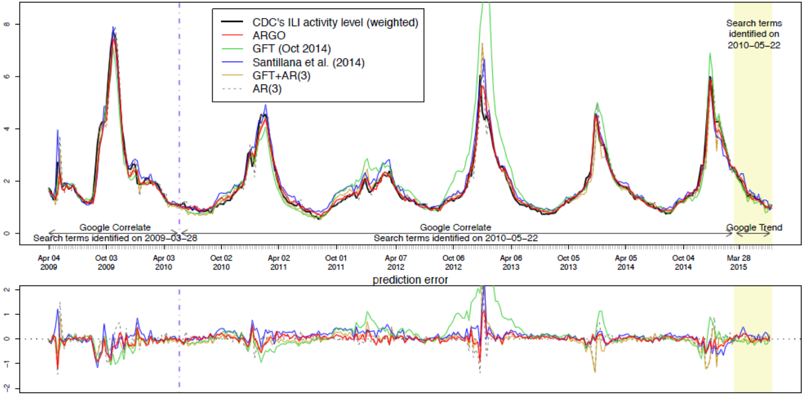
- **Problem:** estimate $\{a_i\}_{i=1}^q$ from $\{y_t\}_{t\geq 0}$; can be formulated as LS
- **Applications:** time-series prediction, speech analysis and coding, spectral estimation...

A Toy Example: the Hang Seng Index Prediction



- blue — the Hang Seng Index during a certain time period.
- red — training, the line is $\sum_{i=1}^{q} a_i y_{t-i}$, and $\mathbf{a}$ is obtained by LS with $q = 10$.
- green — prediction, the line is $\hat{y}_t = \sum_{i=1}^{q} a_i \hat{y}_{t-i}$ with $\mathbf{a}$ obtained in the training phase.

A Real Example: Real-time Prediction of Flu Activity



Tracking influenza outbreaks by ARGO — a model combining the AR model and Google data.

Source: **[Yang-Santillana-Kou15]**.

- **Problem:** given $\mathbf{A} \in \mathbb{R}^{n \times n}$, find a nonzero $\mathbf{v} \in \mathbb{R}^n$ such that

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v}, \qquad \text{for some } \lambda.$$

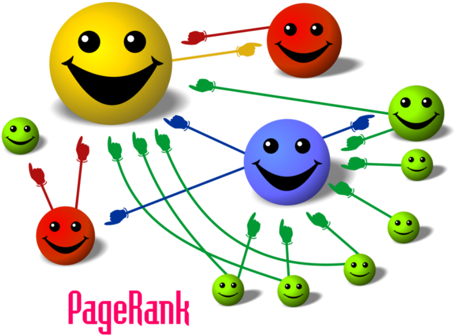
- **Eigenvalue decomposition:** let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be symmetric ($A = A^T$), then it admits a decomposition

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T,$$

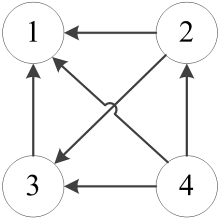
  where $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is orthogonal, i.e., $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$; $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$
- also widely used, either as an analysis tool or as a computational tool
- no closed form in general; can be numerically computed

Application: PageRank

- PageRank is an algorithm used by Google to rank the pages of a search result.
- The idea is to use counts of links of various pages to determine pages' importance.



Source: Wikipedia



- Model:

$$\sum_{j \in \mathcal{L}_i} \frac{v_j}{c_j} = v_i, \quad i = 1, \ldots, n,$$

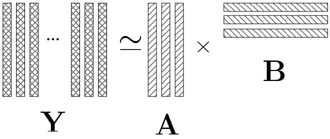
where c_j is the number of outgoing links from page j; $\mathcal{L}_i$ is the set of pages with a link to page i; v_i is the importance score of page i.

- as an example,

$$\overbrace{\begin{bmatrix} 0 & \frac{1}{2} & 1 & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{2} & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 \end{bmatrix}}^{\mathbf{A}} \overbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}}^{\mathbf{v}} = \overbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}}^{\mathbf{v}}.$$

- finding $\mathbf{v}$ is an eigenvalue problem — with n being of order of millions!
- further reading: **[Bryan-Tanya06]**

- **Problem:** given $\mathbf{Y} \in \mathbb{R}^{m \times n}$ and an integer $r < \min\{m, n\}$, find an $(\mathbf{A}, \mathbf{B}) \in \mathbb{R}^{m \times r} \times \mathbb{R}^{r \times n}$ such that either $\mathbf{Y} = \mathbf{AB}$ or $\mathbf{Y} \approx \mathbf{AB}$.



- **Formulation:**

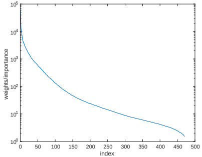
$$\min_{\mathbf{A} \in \mathbb{R}^{m \times r}, \mathbf{B} \in \mathbb{R}^{r \times n}} \|\mathbf{Y} - \mathbf{AB}\|^2,$$

  where $\|\cdot\|$ is the matrix norm.

- **Applications:** model order reduction (MOR), feature extraction, low-rank modeling, ...

Let $\mathbf{Y} \in \mathbb{R}^{m \times n}$ be an image.

original image, sizes 470×641



Store the low-rank factors $\mathbf{A}$, $\mathbf{B}$ instead of $\mathbf{Y}$.

compressed image with $r = 10$

compressed image with $r = 20$

compressed image with $r = 30$

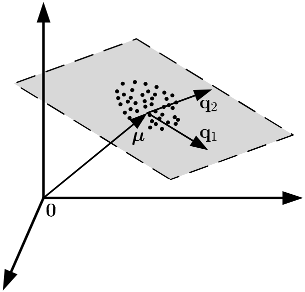
compressed image with $r = 40$

Application II: Principal Component Analysis (PCA)

- **Aim:** given a set of data points $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\} \subset \mathbb{R}^n$ and a positive integer $k < \min\{m, n\}$, perform a low-dimensional representation

$$\mathbf{y}_i = \mathbf{Q}\mathbf{c}_i + \boldsymbol{\mu} + \mathbf{e}_i, \quad i = 1, \ldots, n,$$

where $\mathbf{Q} \in \mathbb{R}^{m \times k}$ is the subspace basis; $\mathbf{c}_i$'s are coefficients; $\boldsymbol{\mu}$ is a base; $\mathbf{e}_i$'s are error vectors.

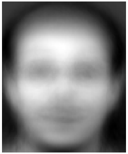
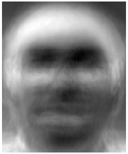
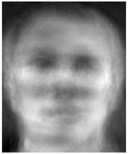
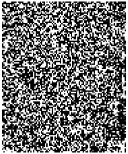


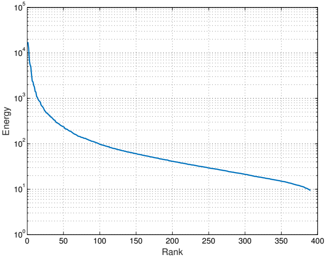
Toy Example: Dimension Reduction of a Face Image Dataset

A face image dataset. Image size $= 112 \times 92$, number of face images $= 400$.

Each $\mathbf{y}_i$ is the vectorization of one face image, leading to $m = 112 \times 92 = 10304$, $n = 400$.

Toy Example: Dimension Reduction of a Face Image Dataset

				
(a) Mean face	(b) 1st principal left singular vector	(c) 2nd principal left singular vector	(d) 3rd principal left singular vector	(e) 400th principal left singular vector



Singular Value Decomposition (SVD)

- **SVD:** Any $\mathbf{Y} \in \mathbb{R}^{m \times n}$ can be decomposed into the following form

$$\mathbf{Y} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

  with

  - $\mathbf{U} \in \mathbb{R}^{m \times m}, \mathbf{V} \in \mathbb{R}^{n \times n}$ are orthonormal matrices
  - $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ is a diagonal matrix with non-negative diagonal entries

- also a widely used analytic and computational tool
- can be numerically computed
- SVD gives optimal solution to the following low-rank approximation problem

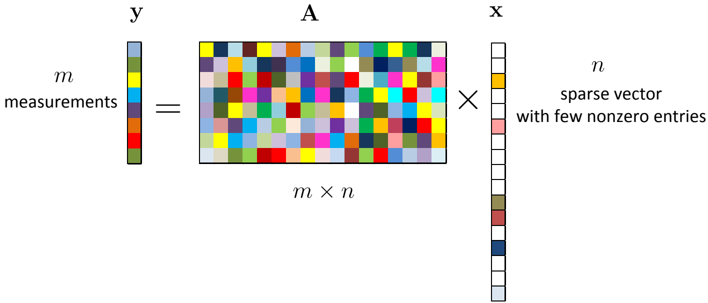
$$\min_{\mathbf{A} \in \mathbb{R}^{m \times r}, \mathbf{B} \in \mathbb{R}^{r \times n}} \|\mathbf{Y} - \mathbf{A}\mathbf{B}\|_2^2.$$

Why Matrix Computations Are Important?

- building blocks and useful tools for
  - electrical engineering
  - signal processing, image processing
  - optimization, machine learning
  - systems and control, dynamical system analysis
- helps you build the foundations for “hot” topics such as
  - sparse recovery;
  - structured low-rank matrix approximation;
  - matrix completion.

Problem: given $\mathbf{y} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $m < n$, find a sparsest $\mathbf{x} \in \mathbb{R}^n$ such that

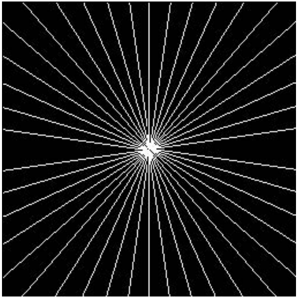
$$\mathbf{y} = \mathbf{A}\mathbf{x}.$$



- by sparsest, we mean that $\mathbf{x}$ should have as many zeros as possible.

Problem: MRI image reconstruction.

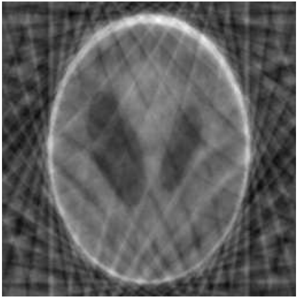
(a) original test image



(b) sampling region in the frequency domain

Source: **[Candès-Romberg-Tao06]**

Problem: MRI image reconstruction.



(a) recovery by filling the unobserved Fourier coefficients to zero

(b) sparse recovery solution

Source: **[Candès-Romberg-Tao06]**

Application: recommendation systems

- in 2009, Netflix awarded \$1 million to a team that performed best in recommending new movies to users based on their previous preference[1] .
- let $\mathbf{Z}$ be a preference matrix, where z_{ij} records how user i likes movie j.

$$\begin{array}{cc} & \text{movies} \\ \mathbf{Z} = & \begin{bmatrix} 2 & 3 & 1 & ? & ? & 5 & 5 \\ 1 & ? & 4 & 2 & ? & ? & ? \\ ? & 3 & 1 & ? & 2 & 2 & 2 \\ ? & ? & ? & 3 & ? & 1 & 5 \end{bmatrix} \text{users} \end{array}$$

  - some entries z_{ij} are missing, since no one watches all movies.
  - $\mathbf{Z}$ is assumed to be of low rank; research shows that only a few factors affect users' preferences.

- **Aim:** guess the unkown z_{ij}'s from the known ones.

[1] www.netflixprize.com

- The 2009 Netflix Grand Prize winners used low-rank matrix approximations **[Koren-Bell-Volinsky2009]**.
- **Formulation** (oversimplified):

$$\min_{\mathbf{A}\in\mathbb{R}^{m\times r},\mathbf{B}\in\mathbb{R}^{r\times n}} \sum_{(i,j)\in\Omega} \left|z_{ij} - [\mathbf{AB}]_{i,j}\right|^2$$

  where Ω is an index set that indicates the known entries of $\mathbf{Z}$.
- cannot be solved by SVD
- in the recommendation system application, it's a large-scale problem
- alternating LS may be used

References

[Yang-Santillana-Kou2015] S. Yang, M. Santillana, and S. C. Kou, "Accurate estimation of influenza epidemics using Google search data via ARGO," *Proceedings of the National Academy of Sciences*, vol. 112, no. 47, pp. 14473–14478, 2015.

[Bryan-Tanya2006] K. Bryan and L. Tanya, "The $25,000,000,000$ eigenvector: The linear algebra behind Google," *SIAM Review*, vol. 48, no. 3, pp. 569–581, 2006.

[Candès-Romberg-Tao2006] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[Koren-Bell-Volinsky2009] B. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42 no. 8, pp. 30–37, 2009.

[Lee-Seung1999] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

Some More Words

- don't be afraid of Math
- understand how people manipulate matrix operations, and how you can use it as a tool;
- know why matrix computations can be performed like in the papers/software
- how to build weapons/tools for your research/problem
- what applications we can do, or to find new applications of our own (learn to apply a tool);
- deep analysis skills (why is this tool valid? Can I invent new tools?)
- feedback is welcome; closed-loop systems are better than open-loop ones