SI231 Matrix Analysis and Computations
Overview

Ziping Zhao

Spring Term 2022–2023

School of Information Science and Technology
ShanghaiTech University, Shanghai, China

http://si231.sist.shanghaitech.edu.cn

Course Information

General Information

- Instructor: Prof. Ziping Zhao
  - office: Rm. 1A-404D, SIST Building
  - e-mail: zhaoziping@shanghaitech.edu.cn
  - website: www.zipingzhao.com
- Lecture hours and venue:
  - Tuesday/Thursday 8:15am–9:55am, Rm. 302, Teaching Center
- Course helpers whom you can consult:
  - TBD
- Course website: http://si231.sist.shanghaitech.edu.cn

Course Contents

- This is a foundation course on matrices, widely used in many different fields, e.g.,
  - machine learning, data mining, computer vision and graphics, natural language processing, information theory, coding theory, cryptography
  - systems and control, signal processing, image and acoustic processing, networking, communications, radar, smart grids, robotics, circuits
  - biomedical engineering, financial engineering, quantum engineering
  - optimization, statistics, econometrics, neural networks, and many more...
- **Aim:** covers matrix analysis and computations at an advanced or research level.
- **Scope:**
  - basic matrix concepts, subspace, norms
  - linear system of equations, LU decomposition, Cholesky decomposition
  - linear least squares, QR decomposition
  - eigendecomposition, positive semidefinite matrices
  - singular value decomposition, pseudo-inverse
  - (advanced) tensor decomposition, advanced matrix calculus, matrix optimization, compressive sensing, structured matrix factorization, matrix applications (especially in machine learning and signal processing), etc.

Learning Resources

- Textbook:
  - Gene H. Golub and Charles F. van Loan, *Matrix Computations* (Fourth Edition), The John Hopkins University Press, 2013.
  - A Chinese translation version by Posts & Telecom Press is available.

- Recommended readings:
  - Roger A. Horn and Charles R. Johnson, *Matrix Analysis* (Second Edition), Cambridge University Press, 2012.
  - Jan R. Magnus and Heinz Neudecker, *Matrix Differential Calculus with Applications in Statistics and Econometrics* (Third Edition), John Wiley and Sons, New York, 2007.
  - Gilbert Strang, *Linear Algebra and Learning from Data*, Wellesley-Cambridge Press, 2019.
  - Giuseppe Calafiore and Laurent El Ghaoui, *Optimization Models,* Cambridge University Press, 2014.

Assessment and Academic Honesty

- Assessment:
  - Assignments: 30%
    * may contain programming questions
    * where to submit: Gradescope
    * 5 assignments in total with 5 “free days” for late submissions.
  - Mid-term examination: 40%
  - Final project: 30%

- Academic honesty:
  - Students are strongly advised to read the ShanghaiTech Policy on Academic Integrity: https://oaa.shanghaitech.edu.cn/2015/0706/c4076a31250/page.htm

Additional Notice

- Sitting in is welcome, and please send the TA your e-mail address to keep you updated with the course.
- You can also get consultation from me; send me an email for an appointment
- Do regularly check your ShanghaiTech (or CAS) e-mail address; this is the only way we can reach you
- The e-learning platform Piazza.com will be used to announce slides and homeworks

Why to Learn the Matrix?

"The Matrix is everywhere. It is all around us. Even now, in this very room."

- Margot Gerritsen, *TEDxStanford: 'The Beauty I See In Algebra'*, 2014.

A Glimpse of Topics in SI231

Linear System of Equations

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

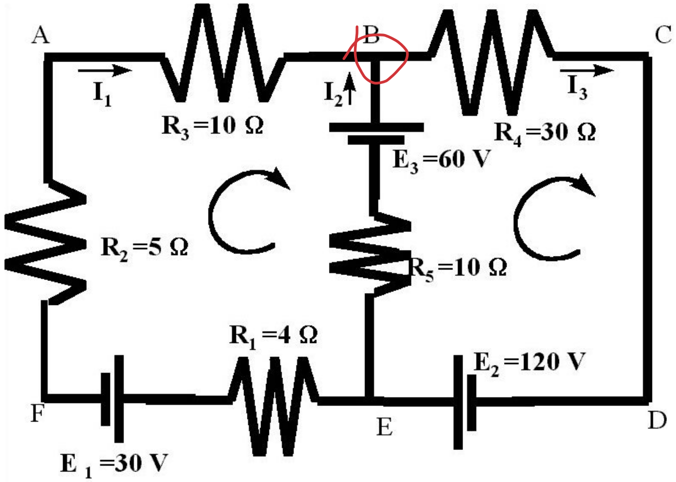
$$\mathbf{A}\mathbf{x} = \mathbf{y}.$$

- **Question 1:** How to solve it?
  - don't tell me answers like `x=inv(A)*y` or `x=A\y` on MATLAB!
  - why the latter is better in terms of execution time and numerical accuracy?
  - this course is about matrix computations

- **Question 2:** How to solve it when n is very large?
  - it's too slow to do the generic trick `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/or $\mathbf{y}$ contain errors?
  - key to system analysis, or building robust solutions

Application Example: Electrical Circuit



- In a given circuit if enough values of currents, resistance, and potential difference is known, we should be able to find the other unknown values of these quantities.
- Mainly use Ohm's Law, Kirchhoff's Voltage Law, and Kirchhoff's Current Law.

$$\overbrace{\begin{bmatrix} 1 & 1 & -1 \\ R_1 + R_2 + R_3 & -R_5 & 0 \\ 0 & R_5 & R_4 \end{bmatrix}}^{\mathbf{A}} \overbrace{\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix}}^{\mathbf{x}} = \overbrace{\begin{bmatrix} 0 \\ -E_1 - E_3 \\ E_2 + E_3 \end{bmatrix}}^{\mathbf{y}}.$$

- Imagine we have a much more complicated circuit network...

Least Squares (LS)

- **Problem:** given $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{y} \in \mathbb{R}^m$, 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, and mathematics

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

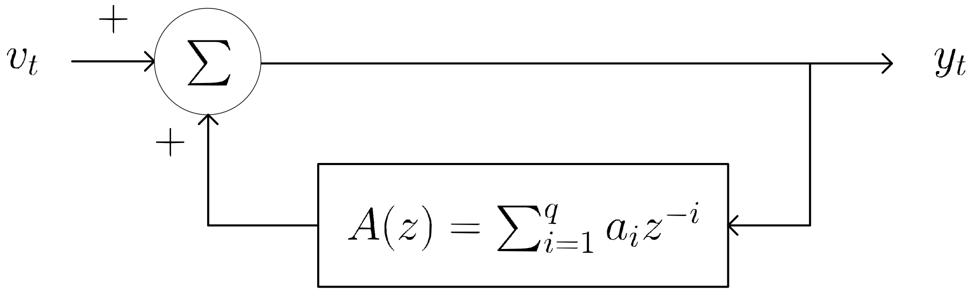
$$\mathbf{x}_{\text{LS}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}.$$

Application Example: Linear Prediction

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

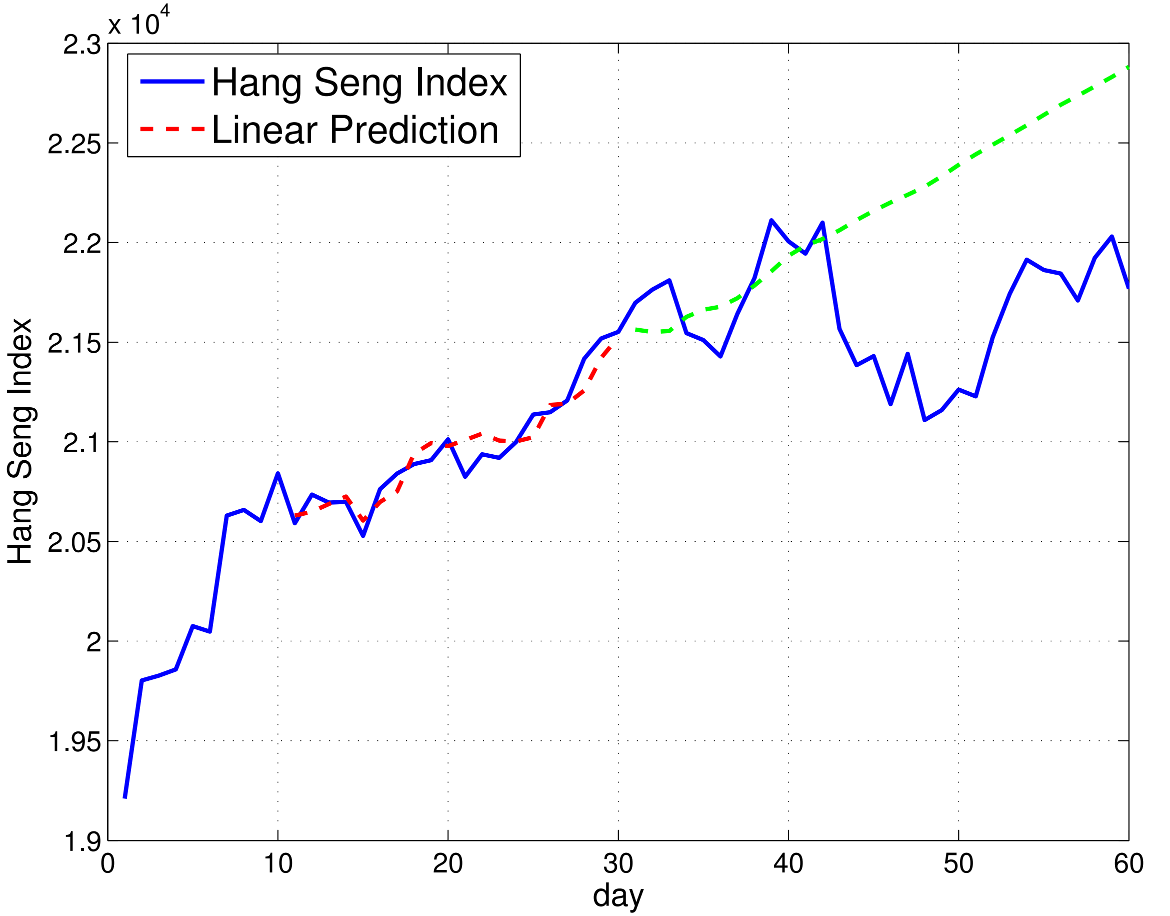
$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_q y_{t-q} + v_t, \quad t = 0, 1, 2, \ldots$$

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 Demo: Predicting Hang Seng Index

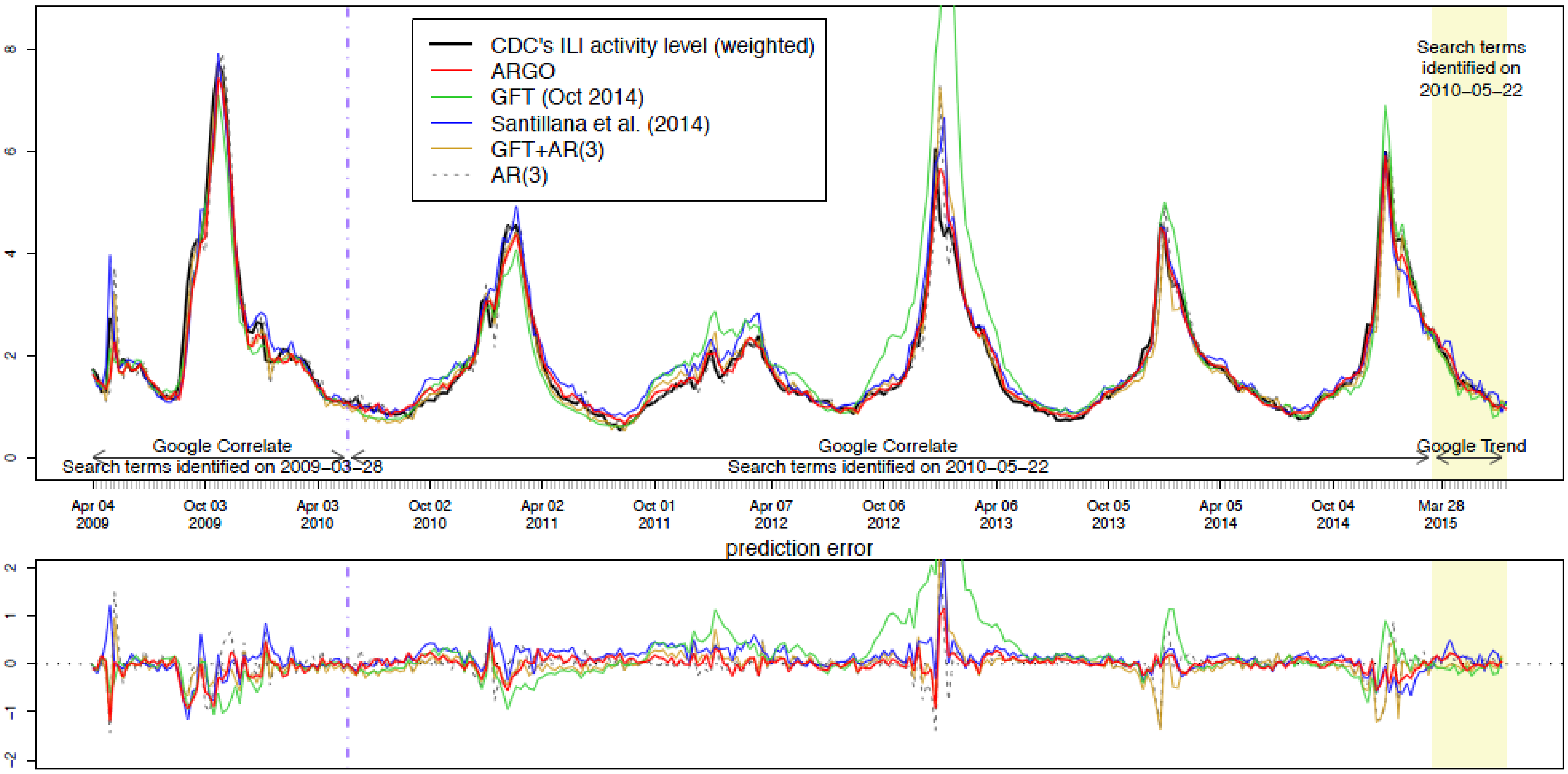


blue— Hang Seng Index during a certain time period.

red— training phase; the line is $\sum_{i=1}^{q} a_i y_{t-i}$; $\mathbf{a}$ is obtained by LS; $q = 10$.

green— prediction phase; the line is $\hat{y}_t = \sum_{i=1}^{q} a_i \hat{y}_{t-i}$; the same $\mathbf{a}$ as 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 search data. Source: **[Yang-Santillana-Kou2015]**.

Eigenvalue Problem

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

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

- **Eigendecomposition:** let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be symmetric; i.e., $a_{ij} = a_{ji}$ for all i, j. It also admits a decomposition/factorization

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

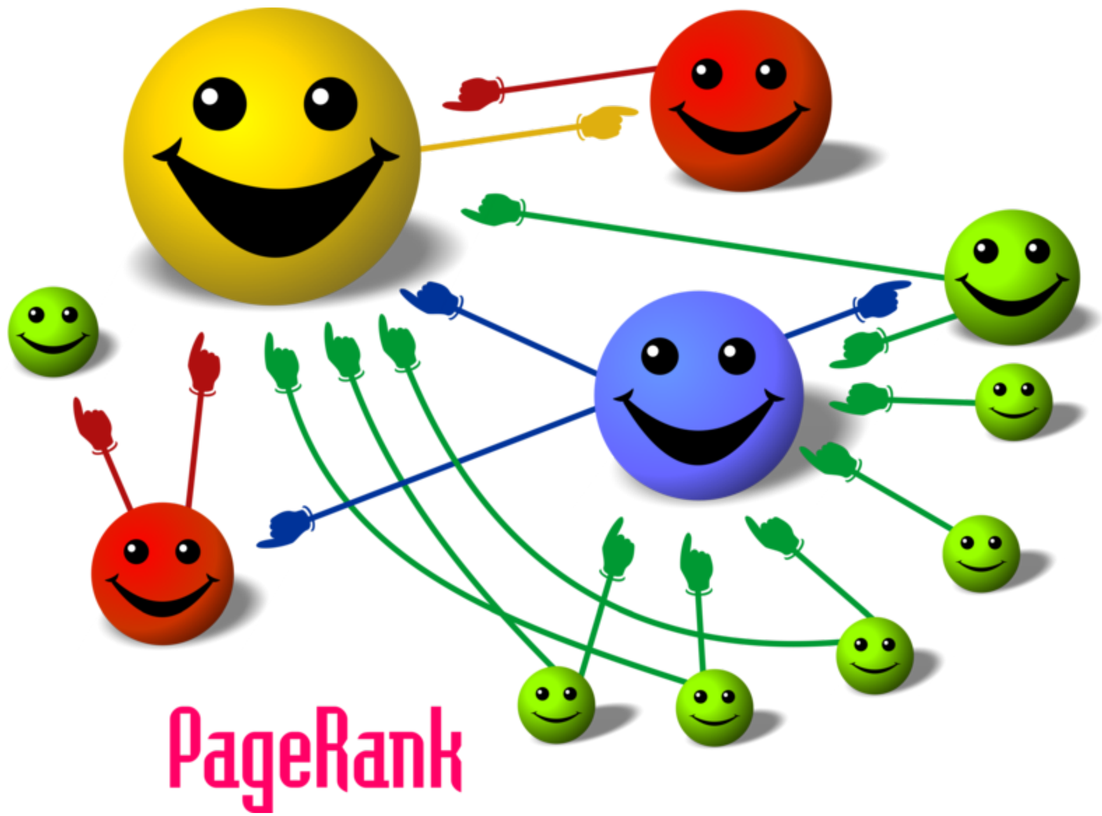
  where $\mathbf{V} \in \mathbb{R}^{n \times n}$ is orthogonal, i.e., $\mathbf{V}^T\mathbf{V} = \mathbf{I}$; $\mathbf{\Lambda} = \mathrm{Diag}(\lambda_1, \ldots, \lambda_n)$

- also widely used, either as an analysis tool or as a computational tool

- no closed form in general; can be numerically computed

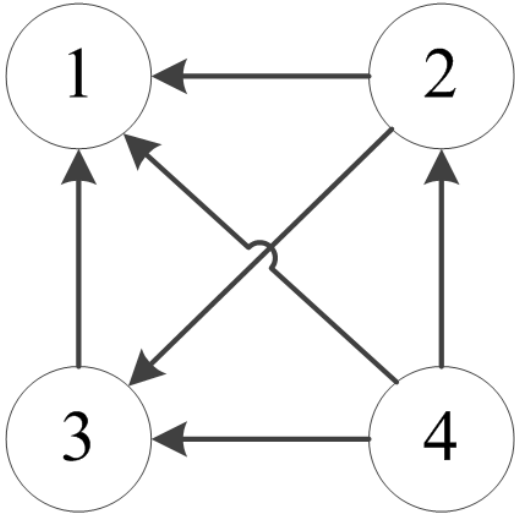
Application Example: PageRank

- 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: Wiki.

One-Page Explanation of How PageRank Works



- 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,

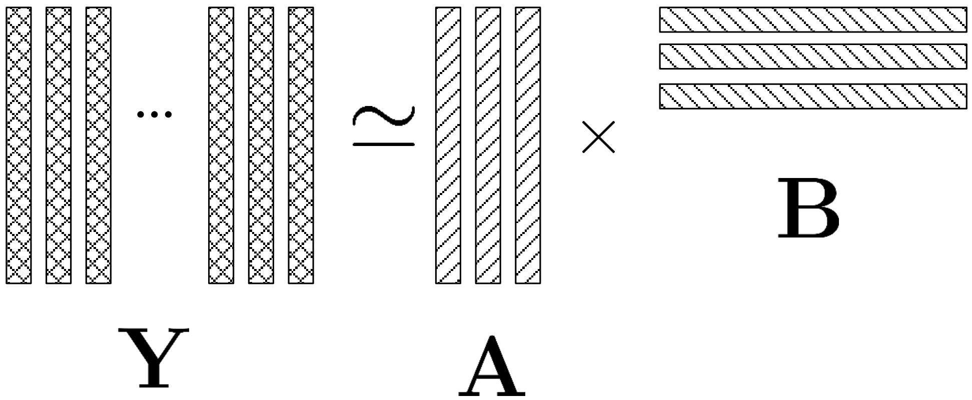
$$\overbrace{\begin{bmatrix} 0 & \frac{1}{2} & 1 & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{2} & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 \end{bmatrix}}^{\mathbf{A}} \overbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}}^{\mathbf{v}} = \overbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\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

- finding $\mathbf{v}$ is an eigenvalue problem—with n being of order of millions!

- further reading: **[Bryan-Tanya2006]**

Low-Rank Matrix Approximation

- **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:**

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

  where $\|\cdot\|_F$ is the Frobenius, or matrix Euclidean, norm.

- **Applications:** dimensionality reduction, extracting meaningful features from data, low-rank modeling, ...

Singular Value Decomposition (SVD)

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

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

  where $\mathbf{U} \in \mathbb{R}^{m \times m}, \mathbf{V} \in \mathbb{R}^{n \times n}$ are orthogonal; $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ takes a diagonal form.

- also a widely used analysis and computational tool; can be numerically computed

- SVD can be used to solve the low-rank matrix approximation problem

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

Application Example: Image Compression

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

original image, size = 101 x 1202

- store the low-rank factor pair $(\mathbf{A}, \mathbf{B})$, instead of $\mathbf{Y}$.

truncated SVD, r = 3

truncated SVD, r = 5

truncated SVD, r = 10

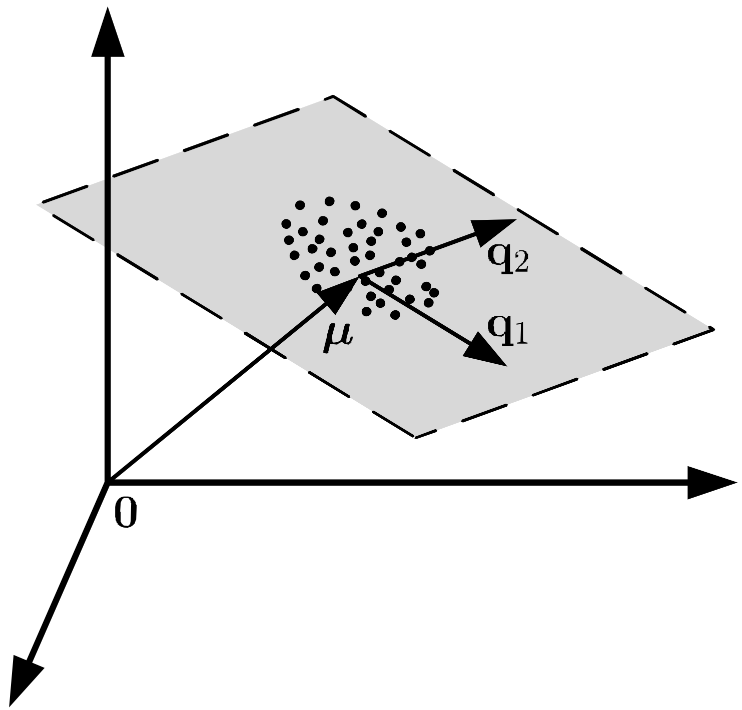
truncated SVD, r = 20

Application Example: Principal Component Analysis (PCA)

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

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

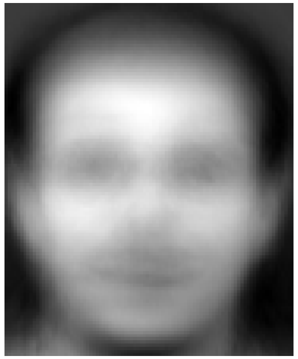
where $\mathbf{Q} \in \mathbb{R}^{m \times r}$ is a basis; $\mathbf{c}_i$'s are coefficients; $\boldsymbol{\mu}$ is a base; $\mathbf{e}_i$'s are errors



Toy Demo: Dimensionality Reduction of a Face Image Dataset

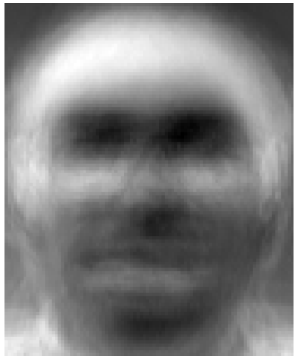
A face image dataset. Image size $= 112 \times 92$, number of face images $= 400$. Each $\mathbf{y}_i$ is the vectorization of one face image, leading to $m = 112 \times 92 = 10304$, $n = 400$.

Toy Demo: Dimensionality Reduction of a Face Image Dataset

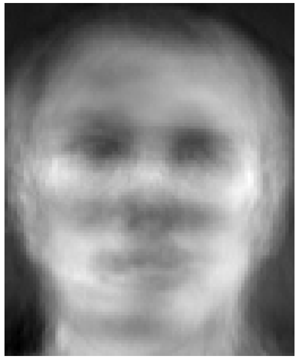


Mean face

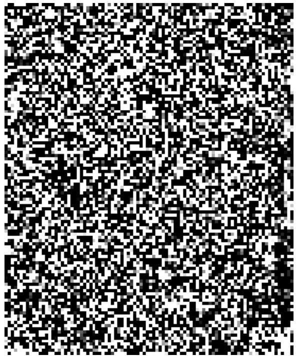
1st principal left singular vector



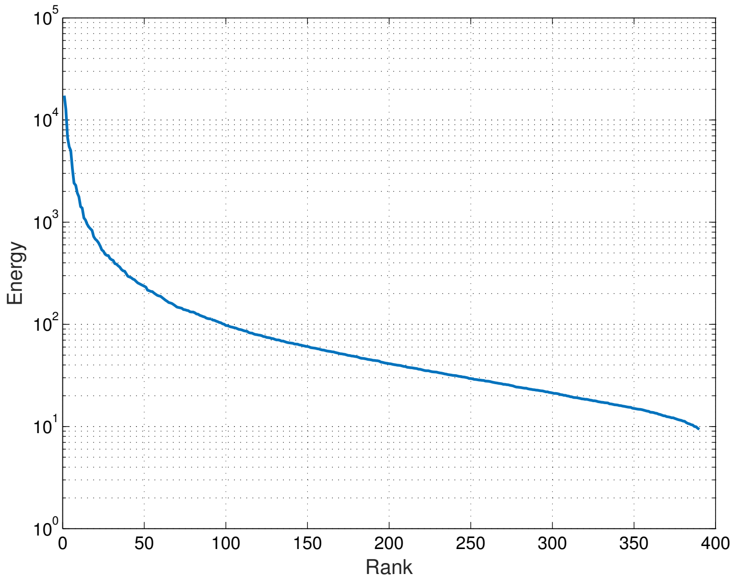
2nd principal left singular vector



3rd principal left singular vector



400th left singular vector



Energy Concentration

Why Matrix Analysis and Computations is Important?

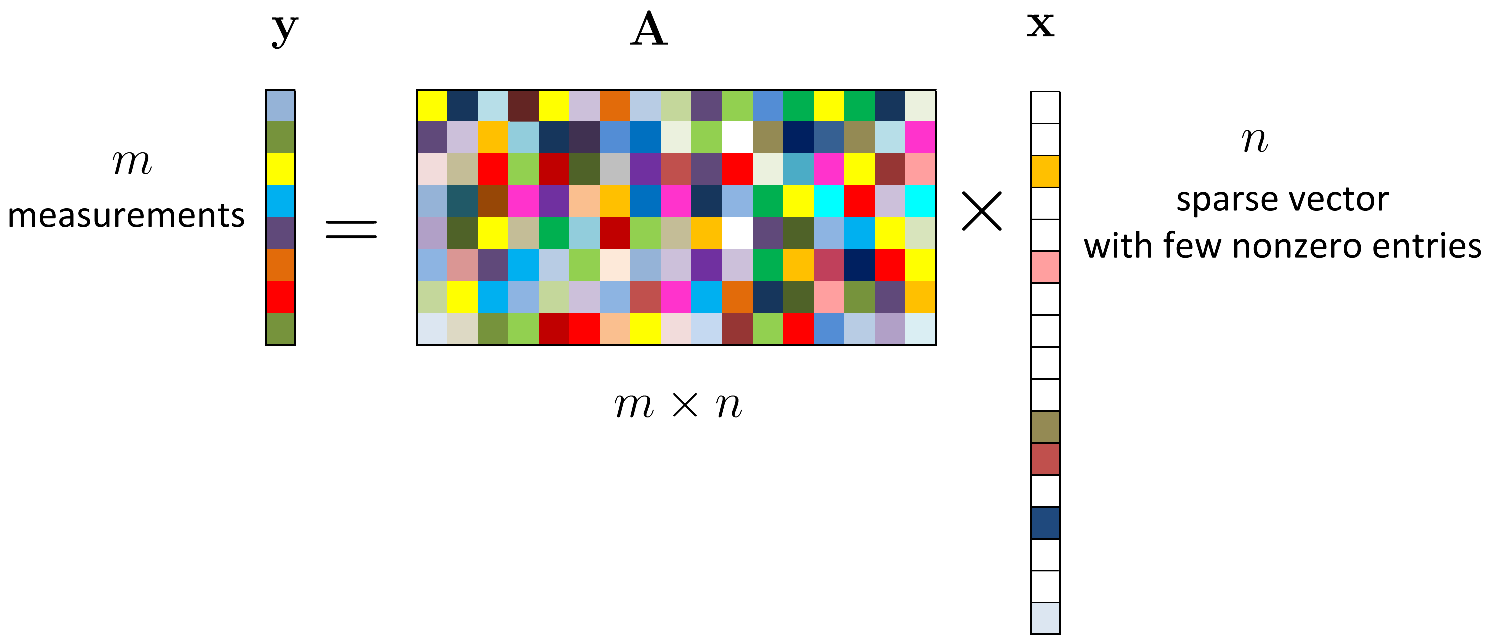
- as said, areas such as signal processing, image processing, machine learning, optimization, computer vision, control, communications, . . ., use matrix operations extensively

- it helps you build the foundations for understanding "hot" topics such as
  - sparse recovery or compressed sensing;
  - matrix completion; structured low-rank matrix approximation; robust PCA;
  - quadratic system of equations problem or phase retrieval;
  - deep neural networks; etc.

The Sparse Recovery Problem

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

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

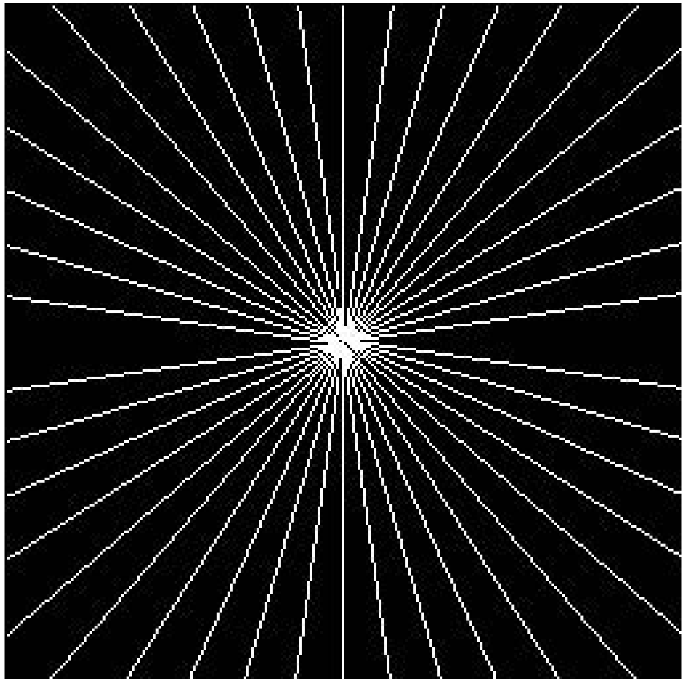


- by sparsest, we mean that $\mathbf{x}$ should have as many zero elements as possible.
- a research topic based on the advances in sparse optimization

Application: Magnetic resonance imaging (MRI)

Problem: MRI image reconstruction.

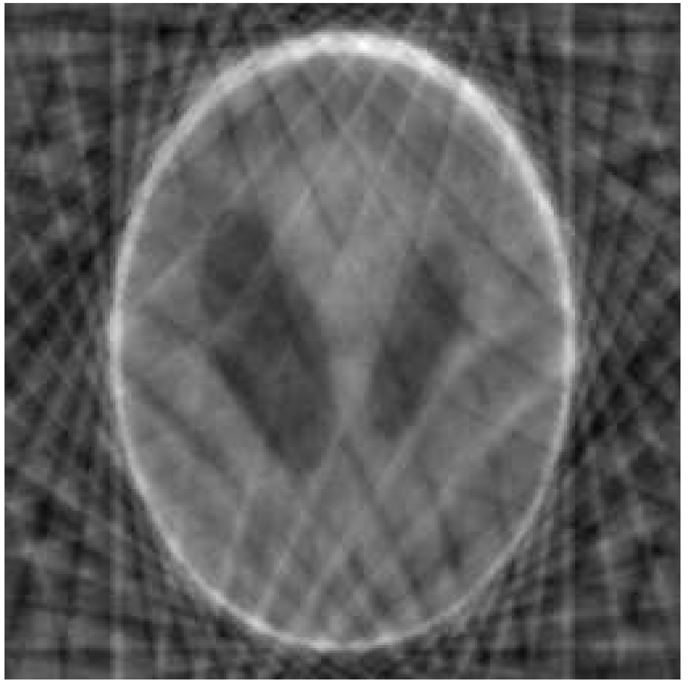
(a)



(b)

Fig. (a) shows the original test image. Fig. (b) shows the sampling region in the frequency domain. Fourier coefficients are sampled along 22 approximately radial lines. Source: **[Candès-Romberg-Tao2006]**

Application: Magnetic resonance imaging (MRI)



(c)

(d)

Fig. (c) is the recovery by filling the unobserved Fourier coefficients to zero. Fig. (d) is the recovery by a sparse recovery solution. Source: **[Candès-Romberg-Tao2006]**

Low-Rank Matrix Completion

- **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.

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

  - 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 unknown z_{ij}'s from the known ones.

[1] www.netflixprize.com

Low-Rank Matrix Completion

- The 2009 Netflix Grand Prize winners used low-rank matrix approximations **[Koren-Bell-Volinsky2009]**.

- **Formulation** (oversimplified):

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

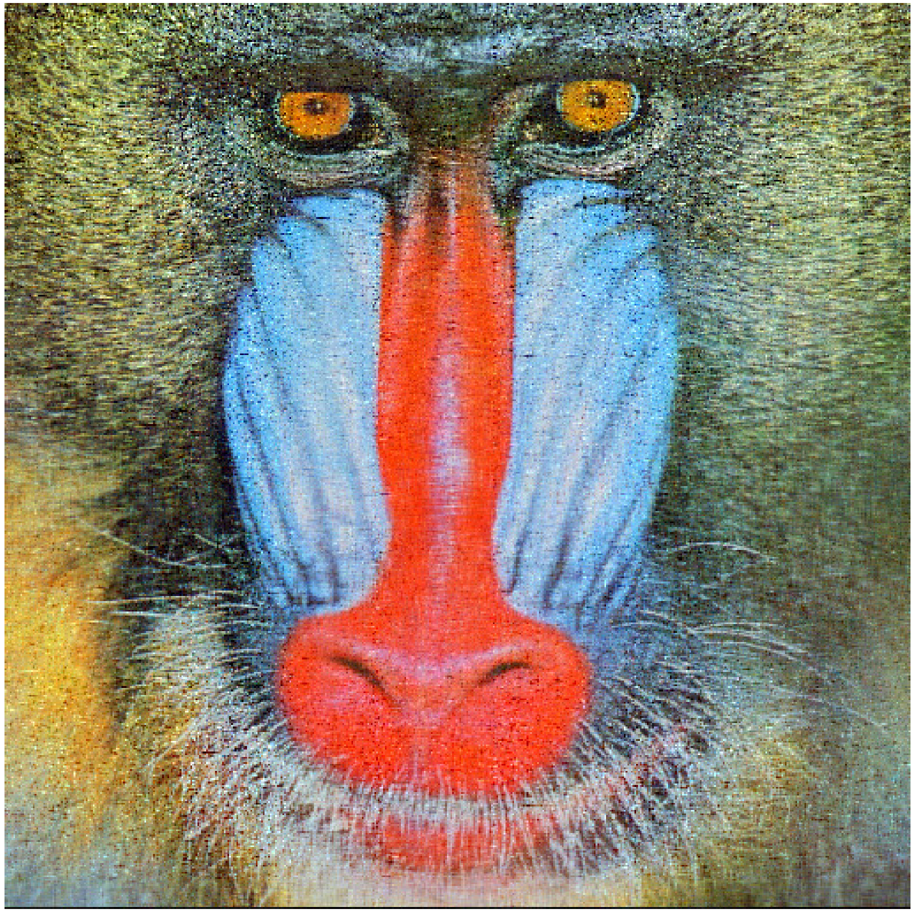
  where Ω is an index set that indicates the known entries of $\mathbf{Z}$.

- cannot be solved by SVD

- in the recommendation system application, it's a large-scale problem

- alternating LS may be used

Toy Demonstration of Low-Rank Matrix Completion



Left: An incomplete image with 40% missing pixels. Right: the low-rank matrix completion result. $r = 120$.

Nonnegative Matrix Factorization (NMF)

- **Aim:** we want the factors to be non-negative
- **Formulation:**

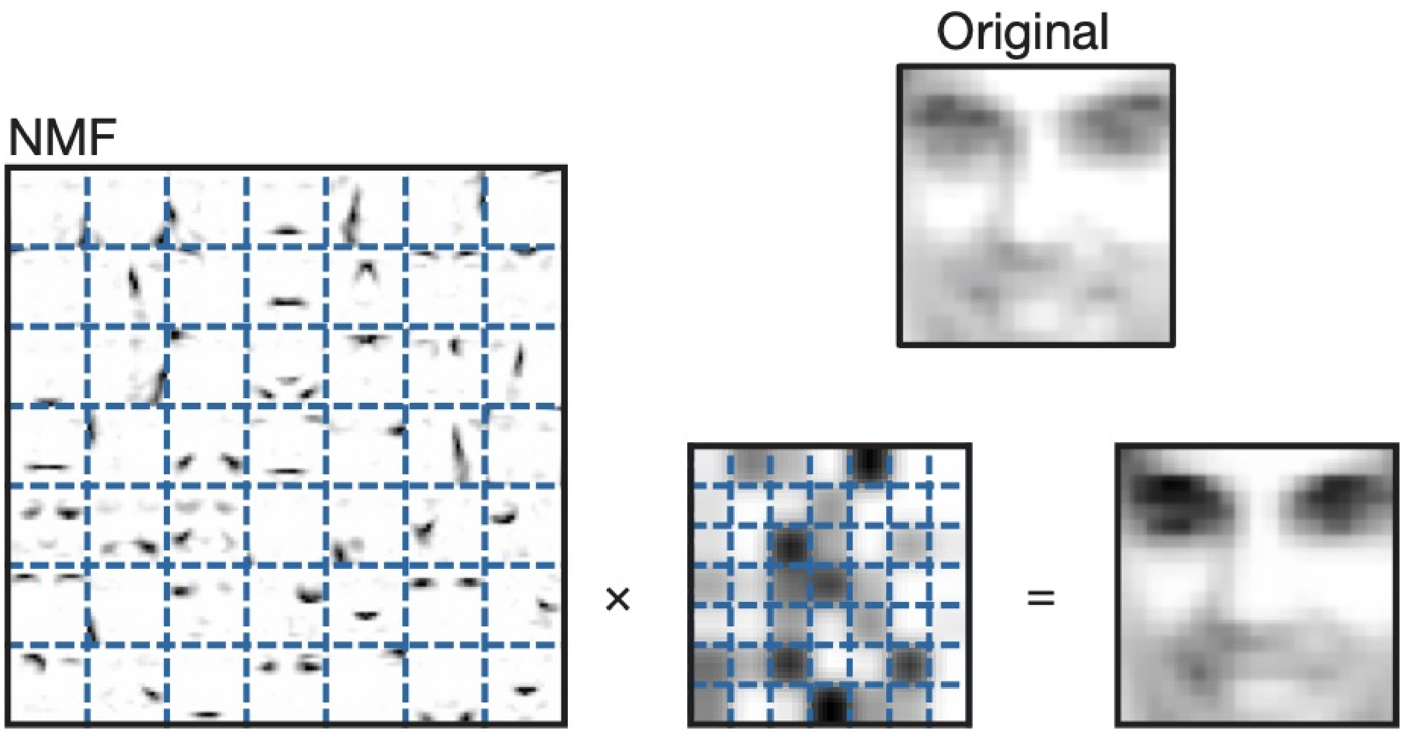
$$\min_{\mathbf{A}\in\mathbb{R}^{m\times r},\mathbf{B}\in\mathbb{R}^{r\times n}} \|\mathbf{Y}-\mathbf{A}\mathbf{B}\|_F^2 \quad \text{s.t. } \mathbf{A}\geq\mathbf{0},\mathbf{B}\geq\mathbf{0},$$

  where $\mathbf{X}\geq\mathbf{0}$ means that $x_{ij}\geq 0$ for all i,j.

- arguably a topic in optimization, but worth noticing
- found to be able to extract meaningful features (by empirical studies)
- numerous applications, e.g., in machine learning, signal processing, remote sensing

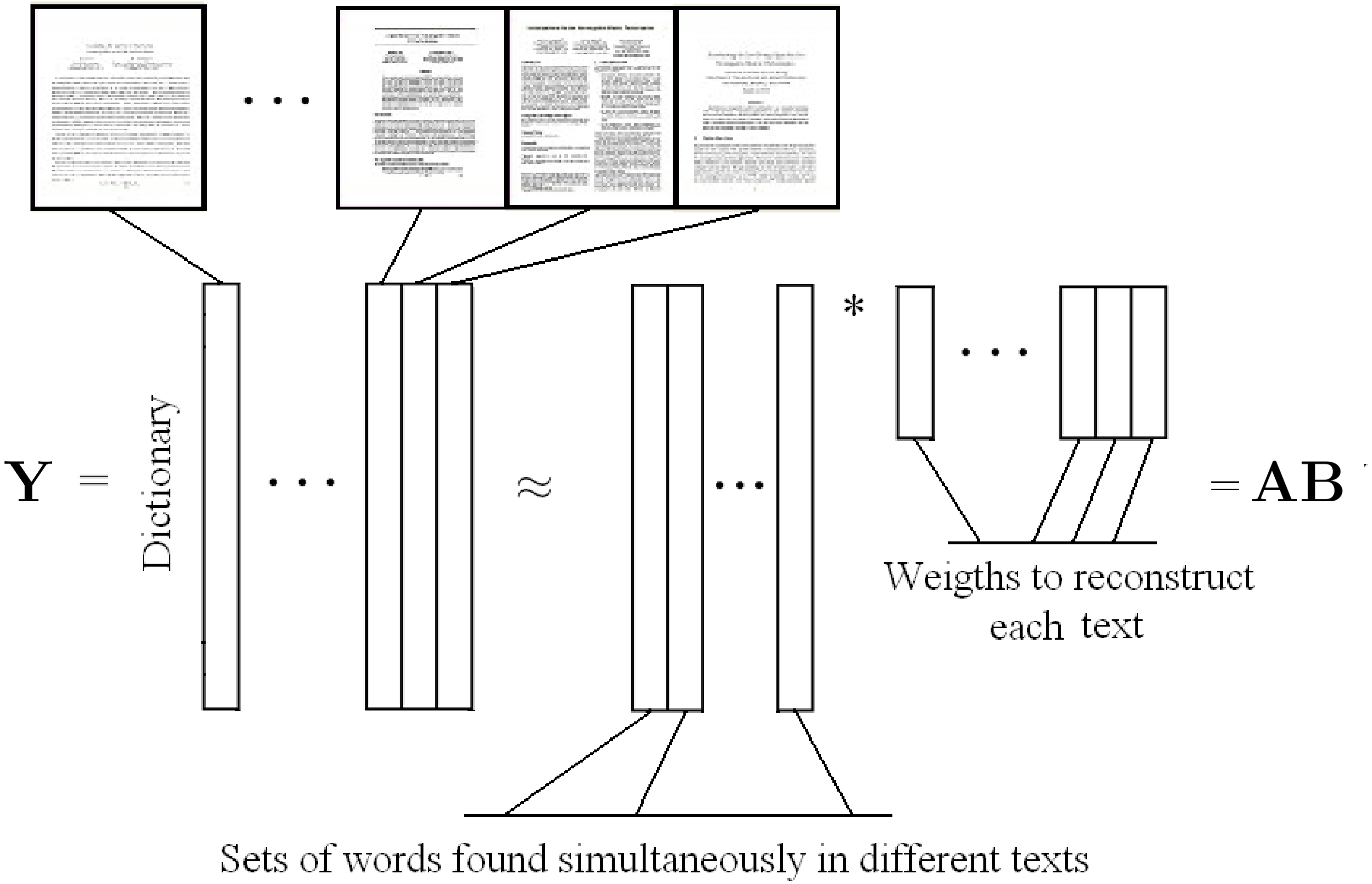
NMF Examples

- **Image Processing:**



The basis elements extract facial features such as eyes, nose and lips. Source: **[Lee-Seung1999]**.

- **Text Mining**:

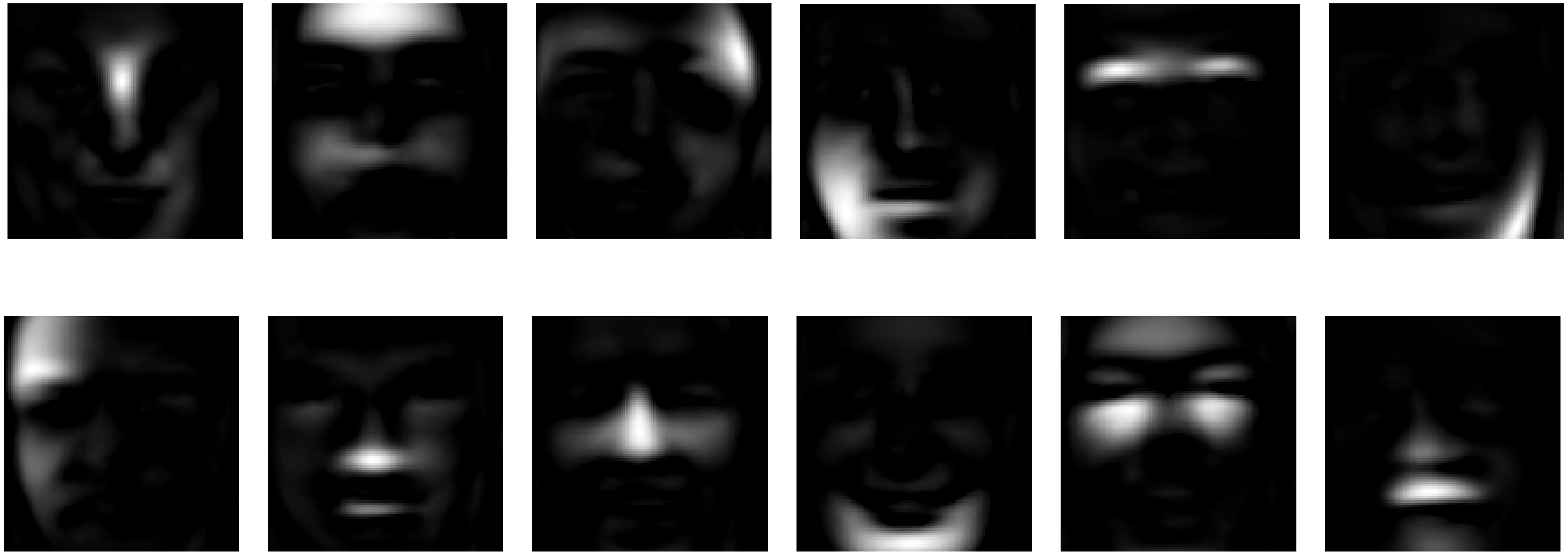


  – basis elements allow us to recover different topics;
  – weights allow us to assign each text to its corresponding topics.

Toy Demonstration of NMF

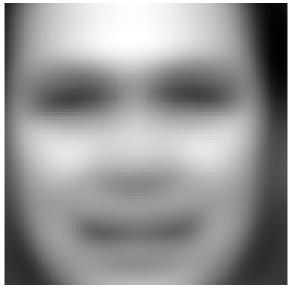
A face image dataset. Image size $= 101 \times 101$, number of face images $= 13232$. Each $\mathbf{y}_i$ is the vectorization of one face image, leading to $m = 101 \times 101 = 10201$, $n = 13232$.

Toy Demonstration of NMF: NMF-Extracted Features

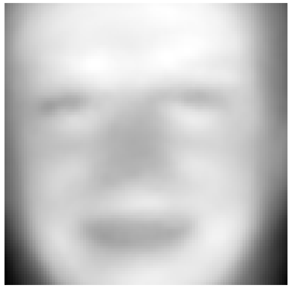


NMF settings: $r = 49$, Lee-Seung multiplicative update with 5000 iterations.

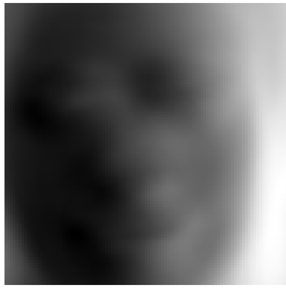
Toy Demonstration of NMF: Comparison with PCA



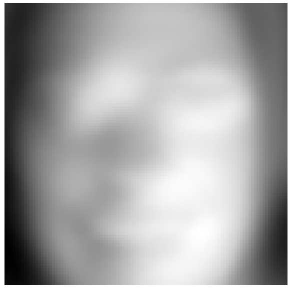
Mean face



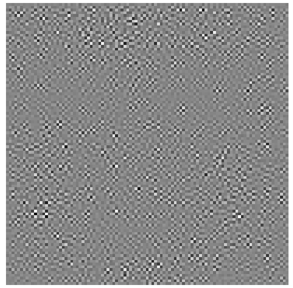
1st principal left singular vector



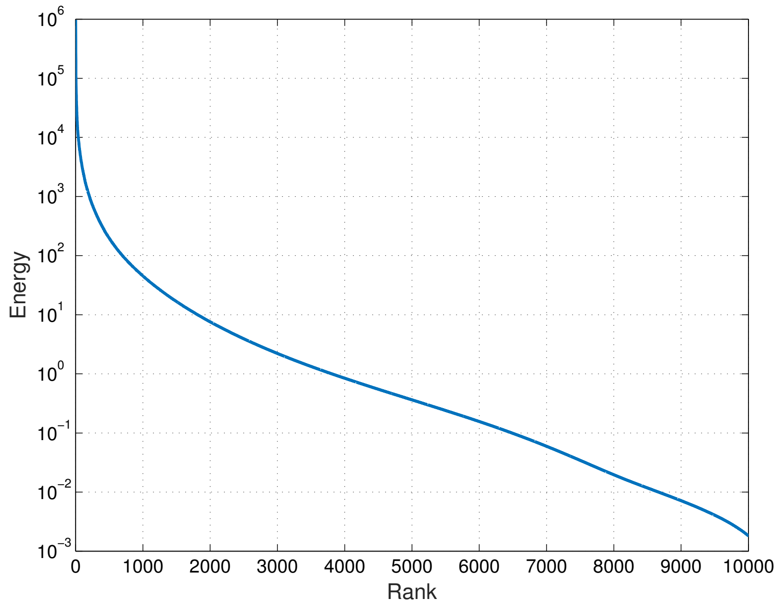
2nd principal left singular vector



3th principal left singular vector



last principal left singular vector



Energy Concentration

A Few More Words to Say

- things I hope you will learn
  - how to read how people manipulate matrix operations, and how you can manipulate them (learn to use a tool);
  - 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? Key to some topics, should go through at least once in your life time)
- critical thinking and active learning, not "passively crammed"
- feedbacks are welcome; closed-loop systems often work better than open-loop

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.