Case Studies D
Singular Value Decomposition (SVD)

Application I: Least Squares

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 \qquad (LS)$$

Here $A \in \mathbb{R}^{m\times n}$.

Assume $\text{rank}(A) = r$.

Let $A = U\Sigma V^T$ be the compact SVD of A, i.e.,

$$U \in \mathbb{R}^{m\times r}, \quad V \in \mathbb{R}^{n\times r}, \quad \Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix} \in \mathbb{R}^{r\times r}.$$

Then, $\min_{x\in\mathbb{R}^n} \|Ax - b\|_2^2 \iff \min_{x\in\mathbb{R}^n} \|U\Sigma V^T x - b\|_2^2$

$$\iff \min_{x\in\mathbb{R}^n} \|U(\Sigma V^T x - U^T b)\|_2^2 + \|(I - UU^T)b\|_2^2$$

$$\iff \min_{x\in\mathbb{R}^n} \|\Sigma V^T x - U^T b\|_2^2 + \|(I - UU^T)b\|_2^2$$

Constant of x

Since Σ is invertible, the minimum is attained when

$$\Sigma V^T x = U^T b,$$

which is solved by $V^T x = \Sigma^{-1} U^T b$.

① When $r = n$, V is square and hence invertible.

Thus, $x = V\Sigma^{-1}U^T b$ is the unique solution to (LS).

② When $r < n$, $V^T x \in \mathbb{R}^r$ and $x \in \mathbb{R}^n$.

Thus, $V^T x = \Sigma^{-1} U^T b$ has infinitely many solutions

It can be checked $x = V\Sigma^{-1}U^T b + y$, where $y \in \ker(V^T)$, is a solution and any solution must be in that form.

Note that $V\Sigma^{-1}U^T b \perp \ker(V^T)$.

So $\|x\|_2^2 = \|V\Sigma^{-1}U^T b\|_2^2 + \|y\|_2^2$

Therefore, the solution of (LS) with a minimum 2-norm is

$$x = V\Sigma^{-1}U^T b$$

we call it the minimun 2-norm solution to (LS)

$A^{\dagger} = V\Sigma^{-1}U^T$ is called pseudo-inverse of A.

When A is invertible, $A^{-1} = A^{\dagger}$

Application II: Sensitivity of linear systems.

Consider $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ is invertible.

In practice, b can not be measured exactly.

Assume we observed $\tilde{b}$. Then we actually solved

$$A\tilde{x} = \tilde{b}$$

what is $\frac{\|\tilde{x}-x\|_2}{\|x\|_2}$ in terms of $\frac{\|\tilde{b}-b\|_2}{\|b\|_2}$?

Let $A = U\Sigma V^T$ be the SVD of A.

Then $x = V\Sigma^{-1}U^T b$ and $\tilde{x} = V\Sigma^{-1}U^T\tilde{b}$.

So
$$\|\tilde{x}-x\|_2 = \|V\Sigma^{-1}U^T(\tilde{b}-b)\|_2$$
$$\le \|V\|_2 \; \|\Sigma^{-1}\|_2 \; \|U^T\|_2 \; \|\tilde{b}-b\|_2$$

(Recall $\|C\|_2$ is the maximum singular value)

$$= 1\cdot \sigma_n^{-1}\cdot 1 \cdot \|\tilde{b}-b\|_2$$
$$= \sigma_n^{-1}\|\tilde{b}-b\|_2.$$

and
$$\|b\|_2 = \|Ax\|_2 = \|U\Sigma V^T x\|_2 \le \|U\|_2 \; \|\Sigma\|_2 \; \|V^T\|_2 \; \|x\|_2$$
$$= 1\cdot\sigma_1\cdot 1\cdot\|x\|_2 = \sigma_1\|x\|_2$$

Therefore,
$$\frac{\|\tilde{x}-x\|_2}{\|x\|_2} \le \frac{\sigma_n^{-1}\|\tilde{b}-b\|_2}{\sigma_1^{-1}\|b\|_2} = \frac{\sigma_1}{\sigma_n}\;\frac{\|\tilde{b}-b\|_2}{\|b\|_2}$$

In other words,

the relative error in the solution x is $\frac{\sigma_1}{\sigma_n}$ times of the relative error in the measured data b.

$\boxed{\text{We call } \kappa = \frac{\sigma_1}{\sigma_n} \text{ the condition number of } A.}$

Larger condition number means more sensitive to the error in b, and the linear system is harder to solve.

For least squares, similar analysis leads to

$$\kappa = \frac{\sigma_1}{\sigma_r}, \text{ where } r \text{ is the rank of } A.$$

This explains why the normal equation method is not preferred in LS.

Because the condition number of $A^TAx = A^Tb$ is

$$\left(\frac{\sigma_1}{\sigma_n}\right)^2, \quad \text{where } \sigma_1, \sigma_n \text{ are max and min singular values of } A.$$

This is the square of the condition number of A, i.e. $\left(\frac{\sigma_1}{\sigma_n}\right)$.

SVD is a fundamental tool in data analysis. The following transforms/tools are essentially SVD:

Principal Component analysis (PCA)

Proper orthogonal decomposition (POD)

Hotelling transform

discrete Karhunen-Loeve (KL) transformation

Application III: Principal Component Analysis (PCA)

Problem: Let $A = [a_1\ a_2 \cdots a_n] \in \mathbb{R}^{m\times n}$ be a dataset matrix, for which each column vector $a_i \in \mathbb{R}^m$ represents a data point

We want to find common features of these points.

For example, we have a dataset of many human faces.

$$A = \begin{bmatrix} | & | & | & \cdots & | & \cdots & | \\ a_1 & a_2 & a_3 & \cdots & a_i & \cdots & a_n \end{bmatrix}$$

$\uparrow$ pixels of a $\sqrt{m} \times \sqrt{m}$ human face image.

We want to find "typical faces" of the n given faces.

Model: Let $x_1, \cdots, x_k \in \mathbb{R}^m$ be k common features of the data set A.

Here k should be significantly smaller than m, n.

We assume each a_i, $i = 1, 2, \cdots, n$, is a linear combination of the common features $x_1, \cdots, x_k$. That is,

$$a_i \approx \sum_{j=1}^{k} c_{ij} x_j, \quad i = 1, 2, \cdots, n,$$

where $c_{ij} \in \mathbb{R}$ is the coefficient of x_j in a_i.

If we define

$$X = [x_1 \; x_2 \; \cdots \; x_k] \in \mathbb{R}^{m \times k}$$

and

$$C = \begin{bmatrix} c_{11} & c_{21} & \cdots & c_{n1} \\ c_{12} & c_{22} & \cdots & c_{n2} \\ \vdots & \vdots & & \vdots \\ c_{1k} & c_{2k} & & c_{nk} \end{bmatrix} \in \mathbb{R}^{k \times n}$$

then,

$$A \approx XC,$$

where $X \in \mathbb{R}^{m \times k}$ and $C \in \mathbb{R}^{k \times n}$ are both unknown.

To find X and C, it is natural to minimize the error of $A \approx XC$. Therefore, we solve

$$\min_{\substack{X \in \mathbb{R}^{m \times k} \\ C \in \mathbb{R}^{k \times n}}} \| A - XC \|_F^2$$

Solution by SVD:

Note that $\operatorname{rank}(XC) \le k$.

Reversely, for any matrix $B \in \mathbb{R}^{m\times n}$ satisfying $\text{rank}(B) \le k$, there must be a factorization (called full rank factorization)

$$B = XC \quad , \text{where} \quad X \in \mathbb{R}^{m\times k} \text{ and } C \in \mathbb{R}^{k\times n}.$$

Therefore,

$$\{XC \mid X \in \mathbb{R}^{m\times k}, C \in \mathbb{R}^{k\times n}\} = \{B \in \mathbb{R}^{m\times n} \mid \text{rank}(B) \le k\}$$

Thus,

$$\min_{\substack{X\in\mathbb{R}^{m\times k}\\ C\in\mathbb{R}^{k\times n}}} \|A - XC\|_F^2$$

$$\Updownarrow$$

$$\min_{\text{rank}(B)\le k} \|A - B\|_F^2.$$

This is exactly the best low-rank approximation of A, whose solution is given by

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T = [u_1\ u_2 \cdots u_k] \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_k \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{bmatrix},$$

where $A = \sum_{i=1}^{\min\{m,n\}} \sigma_i u_i v_i^T$ is the SVD of A.

To get X and C, we just need a decomposition $A_k = XC$ where $X \in \mathbb{R}^{m\times k}$ and $C \in \mathbb{R}^{k\times n}$. There are many such decomposition. In some application, we want X to be orthogonal. Then we can choose

$$X = [u_1\ u_2 \cdots u_k] \qquad C = \begin{bmatrix} \sigma_1 v_1^T \\ \sigma_2 v_2^T \\ \vdots \\ \sigma_k v_k^T \end{bmatrix}$$

Algorithm: ① Compute the top k singular values of A, $\sigma_1, \sigma_2, \cdots, \sigma_k$, and the corresponding singular vectors $u_1, u_2, \cdots, u_k$ and $v_1, v_2, \cdots, v_k$.

② Set $X = [u_1\ u_2 \cdots u_k]$ (called principal component)

and $C = \begin{bmatrix} \sigma_1 v_1^T \\ \sigma_2 v_2^T \\ \vdots \\ \sigma_k v_k^T \end{bmatrix}$

③ For data analysis tasks (such as recognition, classification, clustering, etc), we only need to work on $C \in \mathbb{R}^{k\times n}$, which is significantly smaller than the original data set $A \in \mathbb{R}^{m\times n}$.

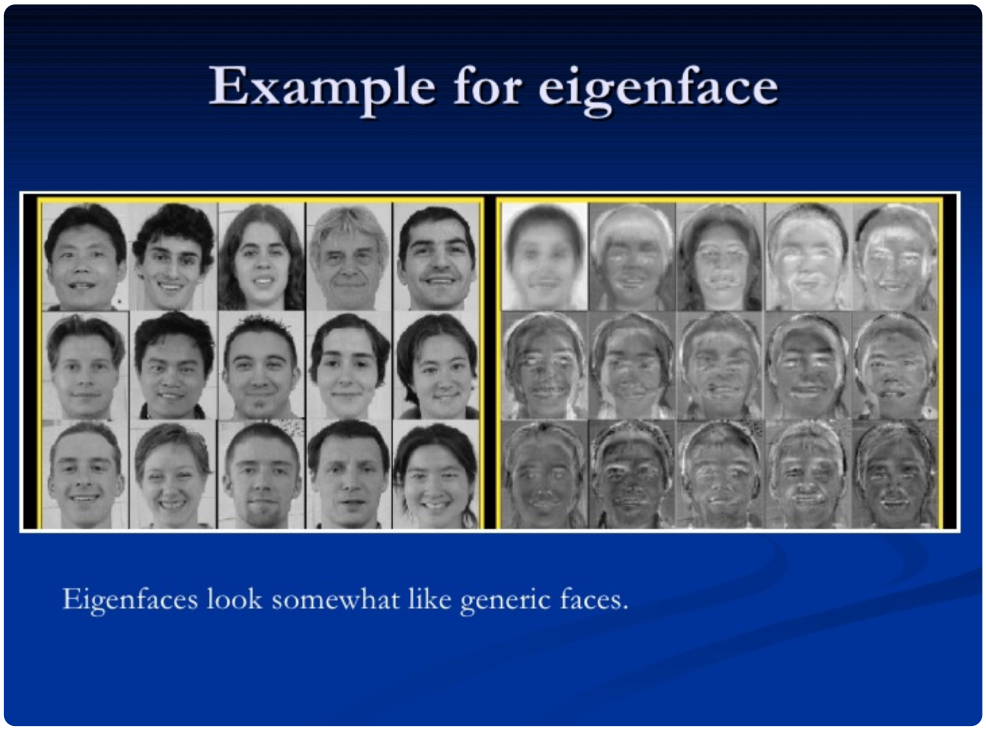
This algorithm is known as principal component analysis (PCA)

Example: Eigenface for face recognition.

Let $A=[a_1\ a_2 \cdots a_n]$, where a_i are human faces.

Then $X=[u_1\ u_2 \cdots u_k]$ are called eigenfaces of the face dataset.

and $C=\begin{bmatrix}\sigma_1 v_1^T \\ \vdots \\ \sigma_k v_k^T\end{bmatrix}$ are the corresponding coefficients.



Then, we only work with the coefficients for data analysis tasks.

For example, for face recognition, if we have a new face $a_{new} \in \mathbb{R}^m$ that is not in A, then we

① Compute the coefficient of a_{new} under the representation of $X=[u_1, u_2, \cdots, u_k]$

$$C_{new} = \begin{bmatrix} u_1^T a_{new} \\ u_2^T a_{new} \\ \vdots \\ u_k^T a_{new} \end{bmatrix}$$

② Write $C=[c_1\ c_2 \cdots c_n]$ where $c_i \in \mathbb{R}^k$. Then, find the closest vector in C to C_{new}, i.e.,

$$i_0 = \arg\min_{1\le i\le n} \|C_{new} - C_i\|_2^2$$

Then, we recognize a_{new} the same face as a_{i_0}.

Application IV: Netflix Prize and Matrix Completion

Netflix Prize: an competition on predicting users ratings to movies, based on previous ratings, held by Netflix.com. The prize was one million US dollars, and given in 2009 to the team which outperformed Netflix's own algorithm by 10%.

Problem and data sets:

100,480,507 ratings that 480,169 users gave to 17,770 movies, in the form of

$$\langle \underset{\substack{\uparrow \\ \text{user id}}}{i}, \underset{\substack{\uparrow \\ \text{movie id}}}{j}, \text{date}, \underset{\substack{\uparrow \\ 1-5 \text{ stars.}}}{\text{rating of user } i \text{ to movie } j} \rangle$$

Matrix model:

Form a rating matrix $M \in \mathbb{R}^{m \times n}$, where m the number of users,
n — — — movies.

and m_{ij} is the rating of user i to movie j.

Then M is incomplete since only a small portion of ratings are known.

2	?	4	?
?	3	?	2
?	3	5	3
3	?	4	?
?	2	?	4

Then the mathematical problem is to complete the matrix:

Find M from m_{ij}, $(i,j) \in \Omega$,

where Ω is a subset and $|\Omega| << mn$.

Assumption: The ratings are only related to r factors. Then

$$m_{ij} = \sum_{k=1}^{r} (\text{rating on factor } k)$$

$$= \sum_{k=1}^{r} \underset{\substack{\uparrow \\ \text{preference of} \\ \text{user } i \text{ to} \\ \text{factor } k}}{p_{ik}} \; \underset{\substack{\uparrow \\ \text{quality of movie } j \\ \text{on factor } k}}{q_{jk}}$$

In matrix form, let

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1r} \\ p_{21} & p_{22} & \cdots & p_{2r} \\ \vdots & \vdots & \vdots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mr} \end{bmatrix} \in \mathbb{R}^{m\times r} \qquad Q = \begin{bmatrix} q_{11} & \cdots & q_{1r} \\ \vdots & \vdots & \vdots \\ q_{n1} & \cdots & q_{nr} \end{bmatrix} \in \mathbb{R}^{n\times r}$$

Then $\quad M = PQ^T. \iff \operatorname{rank}(M) = r \ll \min\{m,n\}.$

Now the problem becomes

Find a rank-r matrix M from $\quad m_{ij},\ (i,j)\in\Omega.$

Solution: Iterative algorithm.

Let $M_0 \in \mathbb{R}^{m\times n}$ be an approximation to M,

which satisfies $\quad (M_0)_{ij} = m_{ij} \quad \forall (i,j)\in\Omega.$

for example, we can choose $\quad [M_0]_{ij} = \begin{cases} m_{ij} & \text{if } (i,j)\in\Omega \\ 0 & \text{if } (i,j)\notin\Omega. \end{cases}$

Given M_k, we generate a better guess M_{k+1} by:

Step 1: Since M is rank-r, we find the best rank-r approximation of M_k. More precisely, define

$$R_k = \arg\min_{\operatorname{rank}(B)\le r} \|M_k - B\|_F$$

By the best low-rank approximation,

$$R_k = \sum_{i=1}^{r} \sigma_i u_i v_i^T, \quad \text{where } M_k = \sum_{i=1}^{\min\{m,n\}} \sigma_i u_i v_i^T \text{ is SVD.}$$

Step 2: Now R_k may not be consitent with the observed entries.

We obtain M_{k+1} by replacing back the known entries.

$$[M_{k+1}]_{ij} = \begin{cases} m_{ij} & \text{if } (i,j)\in\Omega \\ [R_k]_{ij} & \text{if } (i,j)\notin\Omega \end{cases}$$

Then we increase $k \leftarrow k+1$ and repeat the above two steps.