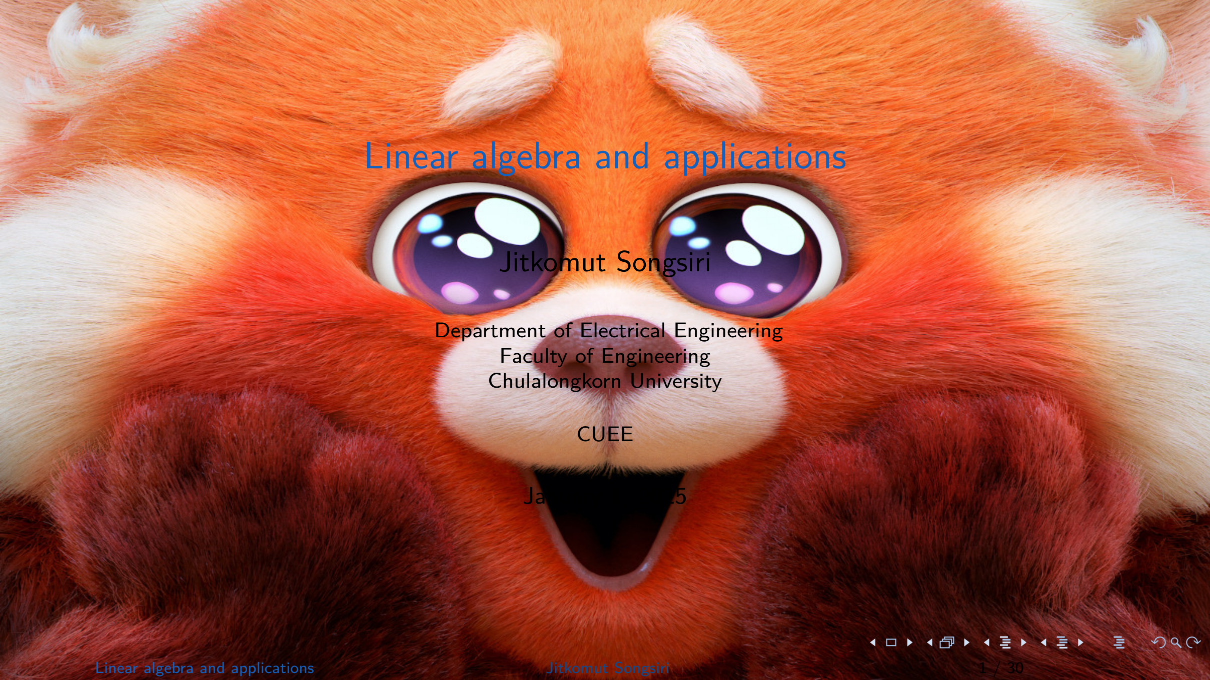


Linear algebra and applications

Jitkomut Songsiri

Department of Electrical Engineering
Faculty of Engineering
Chulalongkorn University

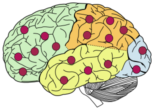
CUEE

Ja[illegible]5

Outline

How to read this handout

1. the note is used with lecture in EE205 (you cannot master this topic just by reading this note) – class activities include
   - graphical concepts, math derivation of details/steps in between
   - computer codes to illustrate examples
2. always read 'textbooks' after lecture
3. pay attention to the symbol ✎; you should be able to prove such ✎ result
4. each chapter has a list of references; find more formal details/proofs from in-text citations
5. almost all results in this note can be Googled; readers are encouraged to 'stimulate neurons' in your brain by proving results without seeking help from the Internet first
6. typos and mistakes can be reported to `jitkomut@gmail.com`

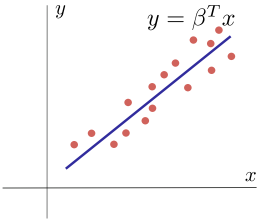


Applications of linear equations

Outline

- least-squares problem
- least-norm problem
- numerical methods in solving linear equations

Least-squares problem



setting: find a linear relationship between y_i and $x_{i,k}$

$$y = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p \triangleq x^T \beta$$

given data as y_i and $x_{i1}, x_{i2}, \ldots, x_{ip}$ for $i = 1, 2, \ldots, N$

the data equation in a matrix form:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} \quad \triangleq \quad y = X\beta$$

problem: given $X \in \mathbf{R}^{m \times n}, y \in \mathbf{R}^m$, solve the linear system for $\beta \in \mathbf{R}^n$

Least-squares: problem statement

overdetermined linear equations:

$$X\beta = y, \quad X \text{ is } m \times n \text{ with } m > n$$

for most y, we cannot solve for β ✎ recall the existence of a solution?

linear least-squares formulation:

$$\underset{\beta}{\text{minimize}} \quad \|y - X\beta\|_2^2 = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} X_{ij}\beta_j - y_i\right)^2$$

- $r = y - X\beta$ is called **the residual error**
- β with smallest residual norm $\|r\|$ is called **the least-squares solution**
- it generalizes solving an overdetermined linear system that cannot be solved *exactly* by allowing the system to have the smallest residual

Least-squares: solution

the zero gradient condition of LS objective is

$$\frac{d}{d\beta}\|y - X\beta\|_2^2 = -X^T(y - X\beta) = 0$$

which is equivalent to the **normal equation**

$$X^T X\beta = X^T y$$

if X is **full rank**, it can be shown that $X^T X$ is invertible:

- least-squares solution can be found by solving the normal equations
- n equations in n variables with a positive definite coefficient matrix
- the closed-form solution is $\beta = (X^T X)^{-1} X^T y$
- $(X^T X)^{-1} X^T$ is the **left inverse** of X

Least-squares: data fitting

given data points $\{(t_i, y_i)\}_{i=1}^N$, we aim to approximate y using a function $g(t)$

$$y = g(t) := \beta_1 g_1(t) + \beta_2 g_2(t) + \cdots + \beta_n g_n(t)$$

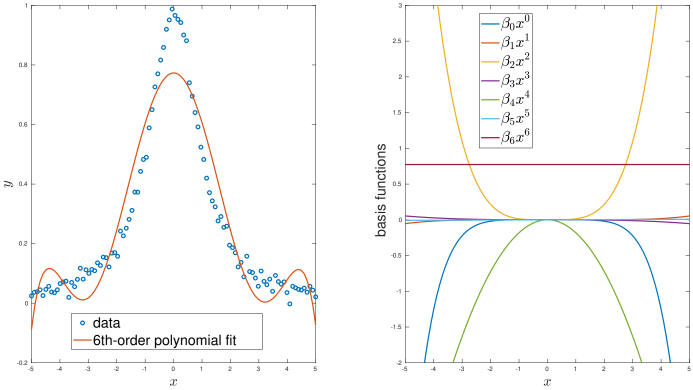
- $g_k(t) : \mathbf{R} \to \mathbf{R}$ is a basis function
  - polynomial functions: $1, t, t^2, \ldots, t^n$
  - sinusoidal functions: $\cos(\omega_k t), \sin(\omega_k t)$ for $k = 1, 2, \ldots, n$
- the linear regression model can be formulated as

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} g_1(t_1) & g_2(t_1) & \cdots & g_n(t_1) \\ g_1(t_2) & g_2(t_2) & \cdots & g_n(t_2) \\ \vdots & & & \vdots \\ g_1(t_m) & g_2(t_m) & \cdots & g_n(t_m) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} \quad \triangleq \quad y = X\beta$$

- often have $m \gg n$, *i.e.*, explaining y using a few parameters in the model

Example

fitting a 6th-order polynomial to data points generated from $f(t) = 1/(1+t^2)$



- (right) the weighted sum of basis functions (x^k) is the fitted polynomial
- the ground-truth function f is nonlinear, but can be decomposed as a sum of polynomials

Least-squares: Finite Impulse Response model

given input/output data: $\{(y(t), u(t))\}_{t=0}^{m}$, we aim to estimate FIR model parameters

$$y(t) = \sum_{k=0}^{n-1} h(k)u(t-k)$$

determine $h(0), h(1), \ldots, h(n-1)$ that gives FIR model output closest to y

$$\begin{bmatrix} y(n-1) \\ y(n) \\ \vdots \\ y(m) \end{bmatrix} = \begin{bmatrix} u(n-1) & u(n-2) & \ldots & u(0) \\ u(n) & u(n-1) & \ldots & u(1) \\ \vdots & \vdots & \vdots & \vdots \\ u(m) & u(m-1) & \ldots & u(m-n+1) \end{bmatrix} \begin{bmatrix} h(0) \\ h(1) \\ \vdots \\ h(n-1) \end{bmatrix}$$

- $y(t)$ is a response to $u(t), u(t-1), \ldots, u(t-(n-1))$
- we did not use initial outputs $y(0), y(1), \ldots, y(n-2)$ since there are no historical input data for those outputs

FIR: example

setting: $y(t+1) = ay(t) + bu(t)$, $y(0) = 0$

- relationship between y and u: write the equation recursively

$$\begin{aligned} y(t) &= a^t y(0) + a^{t-1}bu(0) + a^{t-2}bu(1) + \cdots + bu(t-1) \\ &= a^t y(0) + \sum_{\tau=0}^{t-1} a^{t-1-\tau} bu(\tau) \end{aligned}$$

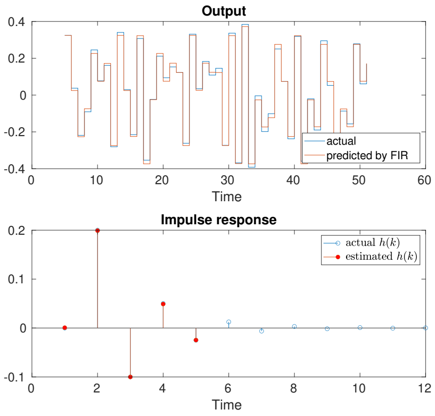
- relate it with the convolution equation: $y(t) = \sum_{k=0}^{\infty} h(k)u(t-k)$

$$h(0) = 0, \quad h(1) = b, \quad h(2) = ab, \quad h(3) = a^2 b, \ldots, \quad h(k) = a^{k-1}b$$

- the actual $h(k)$ decays as k increases but we estimate the first n sequences, *i.e.*, $\hat{h}(0), \hat{h}(1), \ldots, \hat{h}(n-1)$

FIR: example

setting: $a = -0.5, b = 0.2, m = 50, n = 5$, randomize $u(t) \in \{-1, 1\}$

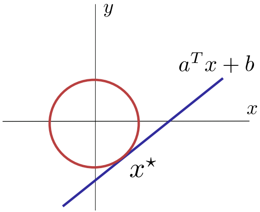


- actual $h(k)$ decays to zero, the first n sequences of $\hat{h}(k)$ are close to actual values
- the predicted output by FIR model is close to the actual output
- $\hat{h}(k)$ is estimated by `A\y` in MATLAB, which returns the least-squares solution

Least-norm problem

setting: given $A \in \mathbf{R}^{m\times n}, b \in \mathbf{R}^m$ where $m < n$ and A is full row rank

(✎ by assumption, the system $Ax = b$ has many solutions)



the least-norm problem is

$$\underset{x}{\text{minimize}} \quad \|x\|_2 \quad \text{subject to} \quad Ax = b$$

- find a point on hyperplane $Ax = b$ that has the minimum 2-norm
- it extends from solving an underdetermined system that has many solutions but we specifically aim to find the solution with smallest norm

Least-norm solution

the least-norm solution is

$$x^\star = A^T(AA^T)^{-1}y$$

- since A is full rank, it can be shown that AA^T is invertible
- $x^\star$ is linear in y and the coefficient is the **right inverse** of A

Proof. let x be any solution to $Ax = b$

- $x - x^\star$ is always orthogonal to x; by using $A(x - x^\star) = 0$

$$(x - x^\star)^T x^\star = (x - x^\star)^T A^T (AA^T)^{-1} y = (A(x - x^\star))^T (AA^T)^{-1} y = 0$$

- $\|x\|$ is always greater than $\|x^\star\|$, hence $x^\star$ is optimal

$$\|x\|^2 = \|x^\star + x - x^\star\|^2 = \|x^\star\|^2 + \underbrace{(x - x^\star)^T x^\star}_{0} + \|x - x^\star\|^2 \geq \|x^\star\|^2$$

Least-norm application: control system

a first-order dynamical system

$$x(t+1) = ax(t) + bu(t), \quad x \text{ is state, } u \text{ is input}$$

problem: given $a, b \in \mathbf{R}$ with $|a| < 1$ and $x(0)$, find

$$\mathbf{u} = (u(0), u(1), \ldots, u(T-1))$$

such that the values of $x(T), x(T-1)$ are as desired and $\mathbf{u}$ has the minimum 2-norm

background: write $x(t)$ recursively, we found that $x(t)$ is linear in $\mathbf{u}$

$$x(t) = a^t x(0) + a^{t-1}bu(0) + a^{t-2}bu(1) + \cdots + bu(t-1) = a^t x(0) + \sum_{\tau=0}^{t-1} a^{t-1-\tau} bu(\tau)$$

Least-norm application: control system

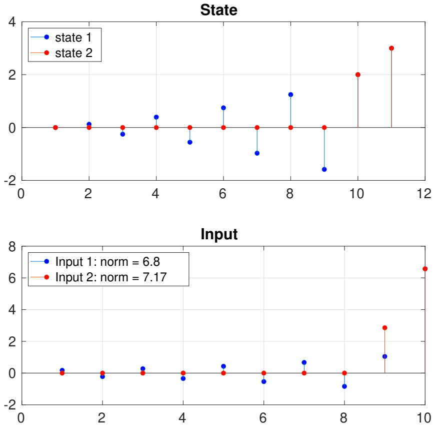
formulate the problem of design $\mathbf{u}$ to drive the state $x(t)$ as desired ✎ verify

$$\begin{bmatrix} x(T) - a^T x(0) \\ x(T-1) - a^{T-1}x(0) \end{bmatrix} = \begin{bmatrix} a^{T-1}b & a^{T-2}b & \cdots & ab & b \\ a^{T-2}b & a^{T-3}b & \cdots & b & 0 \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(T-2) \\ u(T-1) \end{bmatrix} \triangleq y = A\mathbf{u}$$

- regulating the state is a problem of solving an underdetermined system
- A is full row rank, so a solution of $y = A\mathbf{u}$ exists and there are many
- we can try two choices of $\mathbf{u}$:
  1. least-norm solution
  2. any other solution to $y = A\mathbf{u}$

Least-norm application: control system

setting: $a = -0.8, b = 0.7, x(0) = 0, x(T-1) = 2, x(T) = 3$



- different sequences of input drive the state to different paths, but the values of $x(T), x(T-1)$ are as desired
- the least-norm input has the minimum norm – solved by `pinv(A)*y`
- the second choice of input is obtained from `A\y` in MATLAB, which sets many zeros to $\mathbf{u}$ (not the least-norm solution)

Numerical methods in solving linear systems

- solving linear systems by factorization approach
- solving linear systems using softwares
  - square system
  - underdetermined system
  - overdetermined system

Permutation system

a **permutation** matrix P is a square matrix that has exactly one entry of 1 in each row and each column and has zero elsewhere

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

facts: ✎

- P is obtained by interchanging any two rows (or columns) of an identity matrix
- PA results in permuting rows in A, and AP gives permuting columns in A
- $P^TP = I$, so $P^{-1} = P^T$ (simple)
- solving a permuatation system has no cost: $Px = b \Longrightarrow x = P^Tb$

Diagonal system

solve $Ax = b$ when A is diagonal with no zero elements

$$\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

algorithm:

$$\begin{aligned} x_1 &:= b_1/a_{11} \\ x_2 &:= b_2/a_{22} \\ x_3 &:= b_3/a_{33} \\ &\vdots \\ x_n &:= b_n/a_{nn} \end{aligned}$$

cost: n flops

Forward substitution

solve $Ax = b$ when A is lower triangular with nonzero diagonal elements

$$\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

algorithm:

$$\begin{aligned} x_1 &:= b_1/a_{11} \\ x_2 &:= (b_2 - a_{21}x_1)/a_{22} \\ x_3 &:= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \\ &\ \ \vdots \\ x_n &:= (b_n - a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{n,n-1}x_{n-1})/a_{nn} \end{aligned}$$

cost: $1 + 3 + 5 + \cdots + (2n-1) = n^2$ flops

Back substitution

solve $Ax = b$ when A is upper triangular with nonzero diagonal elements

$$\begin{bmatrix} a_{11} & \cdots & a_{1,n-1} & a_{1n} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & a_{n-1,n-1} & a_{n-1,n} \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

algorithm:

$$\begin{aligned} x_n &:= b_n/a_{nn} \\ x_{n-1} &:= (b_{n-1} - a_{n-1,n}x_n)/a_{n-1,n-1} \\ x_{n-2} &:= (b_{n-2} - a_{n-2,n-1}x_{n-1} - a_{n-2,n}x_n)/a_{n-2,n-2} \\ &\vdots \\ x_1 &:= (b_1 - a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n)/a_{11} \end{aligned}$$

cost: n^2 flops

Factor-solve approach

to solve $Ax = b$, first write A as a product of 'simple' matrices

$$A = A_1 A_2 \cdots A_k$$

then solve $(A_1 A_2 \cdots A_k)x = b$ by solving k equations

$$A_1 z_1 = b, \qquad A_2 z_2 = z_1, \qquad \ldots, \qquad A_{k-1} z_{k-1} = z_{k-2}, \qquad A_k x = z_{k-1}$$

complexity of factor-solve method: flops $= f + s$

- f is cost of factoring A as $A = A_1 A_2 \cdots A_k$ (factorization step)
- s is cost of solving the k equations for z_1, z_2, ...z_{k-1}, x (solve step)
- usually $f \gg s$

LU decomposition

for a nonsingular A, it can be factorized as (with row pivoting)

$$A = PLU$$

factorization:

- P permutation matrix, L unit lower triangular, U upper triangular
- **factorization cost**: $(2/3)n^3$ if A has order n
- not unique; there may be several possible choices for P, L, U
- interpretation: permute the rows of A and factor P^TA as $P^TA = LU$
- also known as *Gaussian elimination with partial pivoting* (GEPP)

Not every matrix has an LU factor

without row pivoting, LU factor may not exist even when A is invertible

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \Rightarrow \quad LU = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}$$

from this example,

- if A could be factored as LU, it would require that $l_{11}u_{11} = a_{11} = 0$
- one of L or U would be singular, contradicting to the fact that $A = LU$ is nonsingular

Solving a linear system with LU factor

solving linear system: $(PLU)x = b$ in three steps

- permutation: $z_1 = P^T b$ (0 flops)
- forward substitution: solve $Lz_2 = z_1$ (n^2 flops)
- back substitution: solve $Ux = z_2$ (n^2 flops)

total cost: $(2/3)n^3 + 2n^2$ flops, or roughly $(2/3)n^3$

Softwares (MATLAB)

1. `A\b`
   - square system: it gives the solution: $x = A^{-1}b$
   - overdetermined system: it gives the solution in the least-square sense
   - underdetermined system: it gives the solution to $Ax = b$ where there are K nonzero elements in x when K is the rank of A
2. `rref(A)`: find the reduced row echelon of A
3. `null(A)`: find independent vectors in the nullspace of A
4. `[L,U,P] = lu(A)`: find LU factorization of A

Softwares (Python)

1. `numpy.linalg.solve`: solves a square system (same for scipy)
2. `numpy.linalg.lstsq`: solves a linear system in least-square sense (same for scipy)
3. `sympy.Matrix`: sympy library for symbolic mathematics
4. `scipy.linalg.null_space`: find independent vectors in the nullspace of A
5. `scipy.linalg.lu`: find LU factorization of A

References

1. W.K. Nicholson, *Linear Algebra with Applications*, McGraw-Hill, 2006
2. H.Anton and C. Rorres, *Elementary Linear Algebra*, John Wiley, 2011
3. S. Boyd and L. Vandenberghe, *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least squares*, Cambridge, 2018
4. Lecture notes of EE236, S. Boyd, Stanford
   `https://see.stanford.edu/materials/lsoeldsee263/08-min-norm.pdf`