IAM Data Science Pre-Workshop
Linear Algebra Lecture #1

① Introduction to Linear Systems.

Can arise as linearizations of nonlinear systems. Consider

$$L_1(x_1, x_2) = a_{11}x_1 + a_{12}x_2$$

(a scalar, linear function of two variables).

Notation: $\underline{x} = (x_1, x_2)$, $\underline{a} = (a_{11}, a_{12})$

We will write them as column vectors shortly. Assume $\underline{a}_1 \neq \underline{0}$ for now.

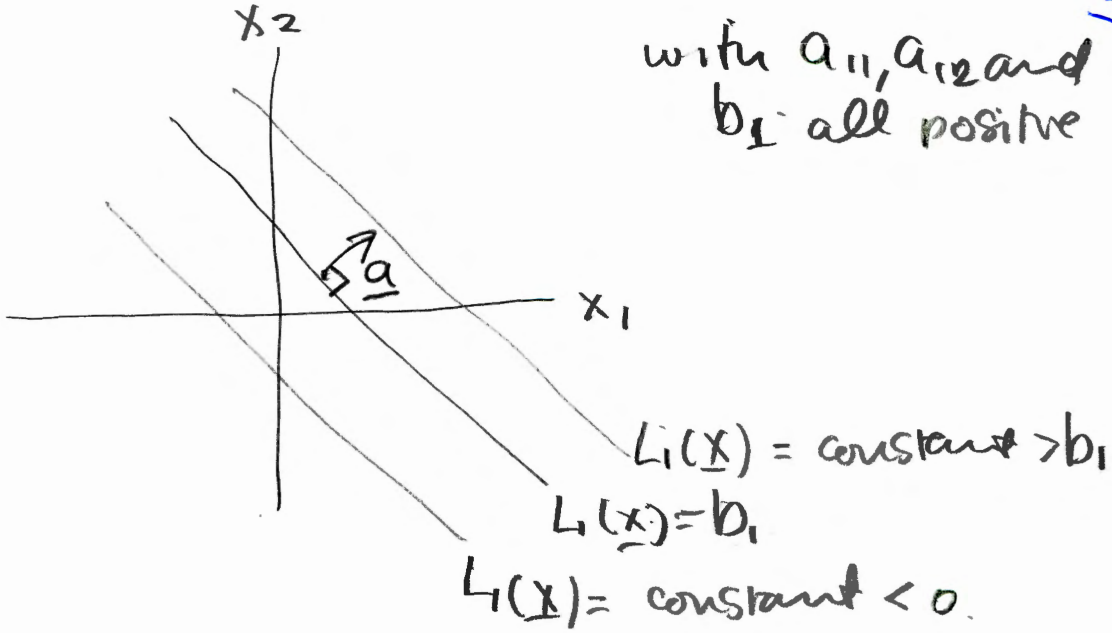
$$\nabla L_1 = (a_{11}, a_{12}) = \underline{a}_1$$

so $\underline{a}_1$ is the normal direction to the level lines of L_1.

Consider $L_1(\underline{x}) = b_1$ in two ways:

(i) Given $\underline{x}$ as input, $L_1(\underline{x})$ gives b_1 as output. Forward model.

(ii) Given b_1 as input, $L_1(\underline{x}) = b_1$ defines a solution set, a line with normal direction $\underline{a}_1$ in this case. Inverse problem.



with a_{11}, a_{12} and b_1 all positive

Now consider a system of m linear equations in n unknowns.

$$\sum_j a_{ij} x_j = b_i$$

$$\begin{matrix} a_{11}x_1 + a_{12}x_2 + \cdots a_{1n}x_1 = b_1 \\ \vdots \qquad\qquad\qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots a_{mn}x_n = b_m \end{matrix} \qquad (A)$$

want to write $\mathbb{A}\,\underline{x} = \underline{b}$ ← $\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$.

$$\mathbb{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ a_{m1} & & a_{mn} \end{bmatrix} \qquad \underline{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

so define matrix multiplication to make this happen.

$$\underset{m \times n}{\mathbb{A}} \; \underset{n \times \ell}{\mathbb{B}} = \underset{m \times \ell}{\mathbb{C}}$$

← ij entry is the inner (dot) product of the i'th row of $\mathbb{A}$ & the j'th column of $\mathbb{B}$.

✓ inner dimensions must match.

interpreted as (i) →.

Looking back at ⊛.

2nd column of A

$$A\underline{x} = x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots x_n \begin{bmatrix} a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix}$$

1st column of A

n'th column of A.

So, $A\underline{x}$ is a linear combination of the column vectors of A.

The set of all linear combinations is called the span of the column vectors of A.

The dimension of the span is called the rank of A (must be less than m).

Consider ⊛ again as rows. Each row defines an n-1 dimensional hyperplane in $\mathbb{R}^n$, if the row vector is nonzero. The span of the row vectors also has the dimension of the rank of A (row rank is the same as column rank).

So the rank of A is $\leq \min(m,n)$. If it is $\min(m,n)$ we say A has full rank.

Consider ⊛ again interpreted as ← (ii). We are looking for the intersection point of m hyperplanes in $\mathbb{R}^n$.

We will restrict ourselves to the case of $m=n$ (square matrix $\mathbb{A}$) for the rest of this lecture.

② Discrete Markov processes (application of (i) →).

Consider a random walk on n states, at integer times α. Let $\underline{x}^{(\alpha)} \in \mathbb{R}^n$ denote the probability vector at time α, i.e.

$x_j^{(\alpha)}$: the probability of being in state j at time α.

If p_{ij} is the probability of moving from state j to state i (note order!), then

$$\underline{x}^{(\alpha+1)} = \mathbb{P}\, \underline{x}^{(\alpha)} \qquad (1).$$

<u>Notes</u>: $p_{ij} \geq 0$, $\sum_{i=1}^{n} p_{ij} = 1$, $\sum_{j=1}^{N} x_j^{\alpha} = 1$ for all α.

columns sum to 1, "you always go somewhere".

p_{ii} can be positive (you remain where you are).

Real probabilists sometimes write this as the transpose and their p_{ij} has the opposite (better) ordering.

The initial state $\underline{x}^{(0)}$ must be given. Then $\underline{x}^{(1)}$ is determined by (1) and $\underline{x}^{(n)}$ recursively.

Note: If $\mathbb{P}$ is "mostly" zeros, we say $\mathbb{P}$ is sparse. This often happens in applications. In this case, (1) can be done efficiently. If P is not sparse (dense), then (1) is an $O(n^2)$ operation count.

Ex Sorcerors' Duel.

Two sorcerors, Ydnew and Xavier, take turns casting spells at each other until one succeeds (Xavier first). Ydnew succeeds 1/2 the time, Xavier 1/3 of the time.

states:
1. No winner, Xavier's turn
2. " " , Ydnew's " .
3. Xavier won
4. Ydnew won.

→ Computational example.

③ Gaussian Elimination (GE algorithm for (ii) $\leftarrow$).

$$\mathbb{A}\underline{x} = \underline{b} \qquad (2).$$

$\mathbb{A}$ $n \times n$ full rank, so (2) has a unique solution $\underline{x}$ for every $\underline{b}$.

Write (2) in an augmented matrix

$$\left[\begin{array}{c|c} \mathbb{A} & \underline{b} \end{array}\right] \qquad (3).$$

If $\mathbb{A}$ were upper diagonal we could easily find the solution by back substitution.

$$\left[\begin{array}{ccc|c} \tilde{a}_{11} & & & \tilde{b}_1 \\ 0 & \tilde{a}_{n-1,n-1} & \tilde{a}_{n-1,n} & \tilde{b}_{n-1} \\ 0 & 0 & \tilde{a}_{nn} & \tilde{b}_n \end{array}\right] \begin{array}{l} \uparrow \text{etc.} \\ \leftarrow x_{n-1} = \dfrac{\tilde{b}_{n-1} - \tilde{a}_{n-1,n}x_n}{\tilde{a}_{n-1,n-1}} \\ \leftarrow x_n = \dfrac{\tilde{b}_n}{\tilde{a}_{nn}} \checkmark \end{array}$$

We can convert (3) to upper triangular form using Gaussian elimination.

$$\left[\begin{array}{ccc|c} a_{11} & \cdots & a_{1n} & b_1 \\ a_{21} & & a_{2n} & b_2 \\ \vdots & & & \\ a_{n1} & & a_{nn} & b_n \end{array}\right]$$

after step 1 these will all be zero.

step 1: replace rows 2 to n by $\text{row}_i - \dfrac{a_{i1}}{a_{11}}\text{row}_1$ does not change the solution set!

Step 2: do the same in column 2 under the diagonal.

Continue until the augmented matrix is in upper triangular form.

Notes: Can be implemented computationally. The implementations include row and column pivoting, recognition of sparsity, and iterative refinement. < more notes p.8

(4) Matrix norms and Condition Number.

Let $\|\cdot\|_*$ be a vector norm on $\mathbb{R}^n$:

i.e. $\|\underline{x}\|_2 = \sqrt{\sum_{j=1}^{n} |x_j|^2}$ Euclidean norm.

$\|\underline{x}\|_\infty = \max_j |x_j|$ maximum norm.

These generate matrix (operator) norms on $n \times n$ matrices

$$\|A\|_* := \max_{\underline{x} \neq \underline{0}} \frac{\|A\underline{x}\|_*}{\|\underline{x}\|_*}.$$

In words, $\|A\|_*$ is the amount that A can increase the size of $\underline{x}$ in the $*$ norm by multiplication.

→ p.9.

If $\mathbb{A}$ is a dense matrix, then solving the system takes $O(n^3)$ operations.

Every $\underline{b}$ gives a unique solution $\underline{x}$ to (2) [if $\mathbb{A}$ has full rank].

The map $\underline{b} \to \underline{x}$ is also linear, so

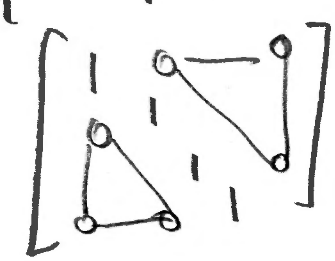
$$\underline{x} = \mathbb{B}\,\underline{b}.$$

call this the inverse of $\mathbb{A}$, $\mathbb{A}^{-1}$.

Finding $\mathbb{A}^{-1}$ and then multiplying by it is more work than just solving the system with GE.

$$\mathbb{A}\mathbb{A}^{-1} = \mathbb{A}^{-1}\mathbb{A} = \text{identity } \mathbb{I}_n.$$

matrix of size n



$\mathbb{I}_n \underline{x} = \underline{x}$ for all $\underline{x}$.

The GE operations can be stored

$$\mathbb{A} = \mathbb{L}\,\mathbb{U}$$

lower triangular

upper triangular.

and then (2) can be solved for other RHS vectors $\underline{b}$ more efficiently, $O(n^2)$ operations.

How sensitive is the solution of

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

to perturbations in $\underline{b}$? That is, how large is $\hat{\underline{x}}$ if

$$A(\underline{x} + \tilde{\underline{x}}) = \underline{b} + \tilde{\underline{b}} \qquad (4)$$

with $\|\hat{\underline{b}}\|_* = \varepsilon \|\underline{b}\|_*$? (small relative error in RHS).

Well $A\underline{x} = \underline{b}$, so $\|\underline{b}\|_* = c\,\|A\|_* \,\|\underline{x}\|_*$ with $c < 1$ but $O(1)$ in general. Apply A^{-1} to (4) [assume we can do this exactly].

$$\underline{x} + \tilde{\underline{x}} = A^{-1}\underline{b} + A^{-1}\tilde{\underline{b}}$$

equal

so $\quad \tilde{\underline{x}} = A^{-1}\tilde{\underline{b}}$

$$\|\tilde{\underline{x}}\|_* \leq \|A^{-1}\|_* \,\|\tilde{\underline{b}}\|_* \leq \varepsilon \underbrace{\|A\|_* \|A^{-1}\|_*}_{\text{condition number of } A,\ K(A)} \|\underline{x}\|_*.$$

So relative errors are increased by a factor of $K(A)$ by the solution process.

$K(A) \geq 1, \quad K(I) = 1.$

(5) Computational Examples.

(a) Approximate $u(x)$, $x \in [0, 2\pi)$, u 2π periodic, that satisfies

$$-u'' + u = f(x) \leftarrow \text{given function.}$$

Use second order finite differences, with $U_j \approx u(jh)$, $j = 0, \dots N-1$ and $h = 2\pi/N$.

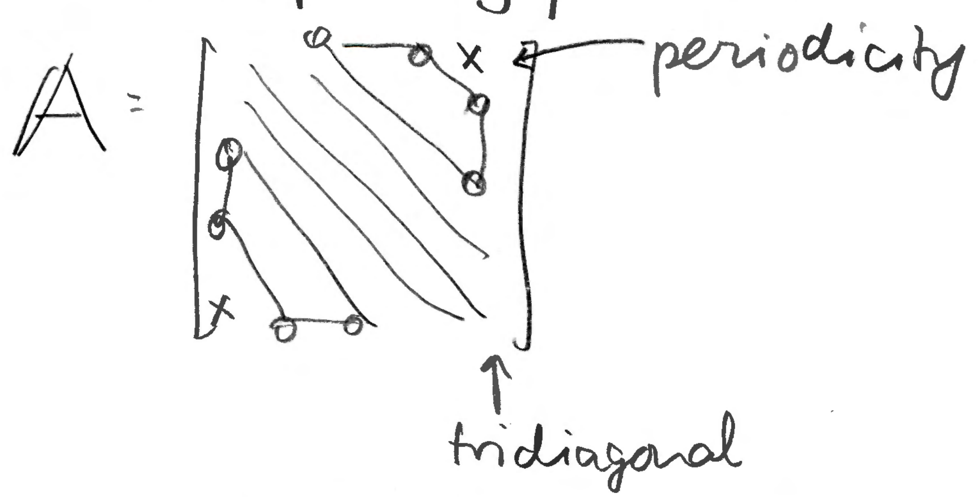
$$\mathbb{A}\,\underline{U} = \underline{F} \leftarrow \begin{bmatrix} f(0) \\ f(h) \\ \vdots \\ f(2\pi - h) \end{bmatrix}. \qquad (5).$$

$$\mathbb{A} = -\mathbb{D}_2 + \mathbb{I}$$

$$(\mathbb{D}_2 \underline{U})_j = \frac{U_{j-1} - 2U_j + U_{j+1}}{h^2}$$

indices mod N.

$\mathbb{A}$ has the sparsity pattern



(5) can be solved with a direct solver with $O(N)$ operations.

(b) approximate $u(x,y)$, $(x,y) \in [0,2\pi)^2$ doubly periodic that satisfies

$$-\Delta u + u = f(x,y) \leftarrow \text{given.}$$

$N \times N$ grid, $n = N^2$ unknowns. Direct, sparse solve $O(N^3) = O(n^{3/2})$ operations.

For (a) & (b) the resulting matrices A are symmetric, positive definite with condition number $O(N^2)$.

The conjugate gradient method can be applied. This is an iterative method. at each step, one multiplication by A and inner products are needed. To converge to an error of size ε,

$$\sqrt{\kappa}\,|\ln \varepsilon| \qquad (6).$$

iterations are needed. For (b), each operation takes $O(N^2)$ work (multiplication by A), and (6) with $\kappa = O(N^2)$ means $O(N)$ operations, total $O(N^3)$ operations for a fixed accuracy. This is the same operation order as the direct solver. In 3D, CG becomes advantageous.

If A is not symmetric, positive definite, there are relatives of CG that can be used in the family of Krylov subspace methods, although the computational complexity is increased.

→ computational examples.

(6) Suggested problems.

#1. Adjust the sorcerors' duel to allow each to have an ablative shield (that can absorb one spell hit). This can be described by a Markov process with 10 states. How often does Xavier win this duel?

Adjust the spell success probabilities so that each wins 1/2 the time and the average duel lasts 9 rounds.

#2. Let A be the 1D FD approximation matrix from section (5). Compute A^{-1} and observe it is a dense matrix.

#3. Develop a fourth order, wide stencil approximation for the 1D & 2D model problems.

#4. Develop a spectrally accurate approximation to the 1D model problem. The resulting matrix will be dense.

#5 Develop an approximation to the equivalent 3D problem. Compare the computational times for the direct vs CG solvers.