“Special” Matrices
LU Factorization

CS 111: Introduction to Computational Science

Spring 2019 Lecture #3

Ziad Matni, Ph.D.

Administrative

- Homework #1 was due on Monday
- Homework #2 due next Monday

Lecture Outline

- Some "special" matrices and related operations
  - Python demonstration
- Exercise: Writing code for matrix multiplication
- A = LU Factorization
  - Using pivots
  - Not using pivots
  - Coding the process

Identity Matrix

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

What’s A.I?

What’s I.A?

Diagonal Matrix

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \qquad A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

What's A.D?

What's D.A? (see blackboard)

Permutation Matrices

P = I matrix but rearranged rows

Example: a 3x3 **P** could be: $\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- If I multiply **P.A**:

  I rearrange the rows of A

- If I multiply **A.P**:

  I rearrange the columns of A

Transpose Operation on Matrices

- $\mathbf{A^T}$: Transpose the rows in A into columns in A

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \text{then} \quad A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

- So, if A is a size `mxn` matrix, then A^T is `nxm`

- If $A = A^T$, then we say that A is a **symmetrical matrix**.

Invertible Matrices

- A **square** matrix A is **invertible** (aka = nonsingular) IFF:
  - det(A) is not zero ← *what is det(A)?*
  - If det(A) = 0, then A is called **singular** and cannot be inverted

- So then we can say that a matrix $\mathbf{A^{-1}}$ exists such that: $\mathbf{AA^{-1} = I}$
- Inverse matrix properties:

$$(A^{-1})^{-1} = A \qquad \det(A^{-1}) = (\det(A))^{-1}$$

$$(A^T)^{-1} = (A^{-1})^T \qquad (AB)^{-1} = B^{-1}A^{-1}$$

Python Demonstration

- Identity matrix
- Diagonal matrices
- Changing single entries, rows, columns
- Permutation matrices

Iterative (naïve) Algorithm for Matrix Multiplication

For: A (n x m) and B (m x p), then C = AB (n x p) can be defined as:

$$c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}.$$

- Input: matrices A and B
- Let C be a new matrix of the appropriate size
- For i from 1 to n:
  - For j from 1 to p:
    - Let $\mathrm{sum} = 0$
    - For k from 1 to m:
      - Set $\mathrm{sum} \leftarrow \mathrm{sum} + A_{ik} \times B_{kj}$
    - Set $C_{ij} \leftarrow \mathrm{sum}$
- Return C

Matrix Factorization

- Recall the Ax = b example where $\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$ $\begin{array}{cc} a00 & a01 \\ a10 & a11 \end{array}$
- Note what happens when we try to do Gaussian elimination on A:
  - In order to make element a_{10} become zero, then I need to make:
    row1 = row1 − C.row2 where: C = −0.5
- Note that I can write **A** as:

$$\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ 0 & 1.5 \end{bmatrix}$$

Note: This matrix has ***1*** *in the diagonals and the bottom part of it has the coefficient* ***C (-0.5)***

- Rule: I can ONLY do this if **A** is invertible!

Your To-Dos

- Sections on Thursday
- Homework #2 – due **Monday, April 15th**

</LECTURE>