Linear Algebra

Kenechi Dukor

Agenda

1. Introduction
2. The Basics
3. Vectors
4. Matrices
5. Solving Linear Equations
6. Sparse Matrices
7. Conclusion

Introduction

- Linear algebra is the branch of mathematics concerning linear equations, linear functions, and their representations in vector spaces and through matrices. It is a unifying thread of almost all mathematics and includes everything from solving elementary equations to developing abstractions such as groups, rings, and fields.

$$a_1x_1 + \ldots + a_nx_n = b$$

In vector notation we say $a^T x = b$. This is the linear transformation of x.

Linear algebra is fundamental to geometry, for defining objects such as lines, planes, rotations.

Key Points:

- Fundamental to Engineering, Physics, and Computer Science.
- Provides tools to solve systems of linear equations.
- Basis for various algorithms in machine learning and data science.

Introduction

Importance in Various Fields:

Computer Graphics:

- Transformation of 3D models into 2D views.
- Manipulation of images and colors using matrices.

Machine Learning:

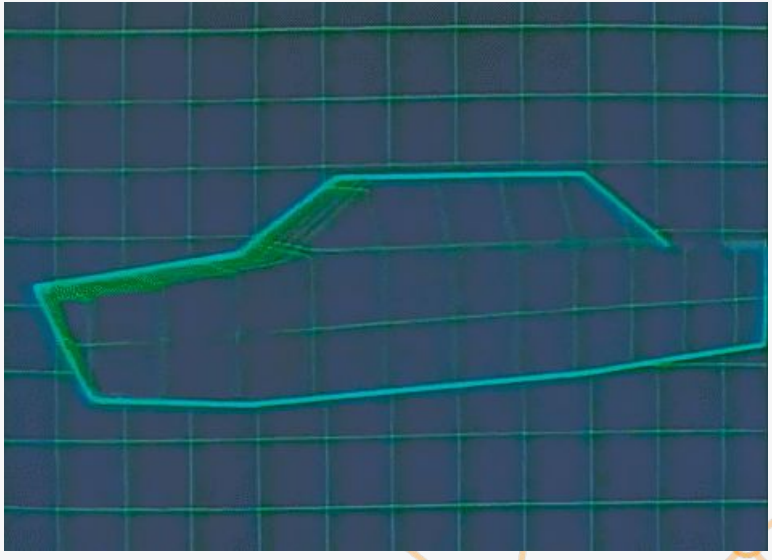
- Data representation and feature extraction.
- Optimization problems, like gradient descent.

Engineering:

- Analysis of systems and structures.
- Electrical circuits, control systems, and structural engineering.

Others:

- Economics for input-output models.
- Quantum mechanics for state transformations.

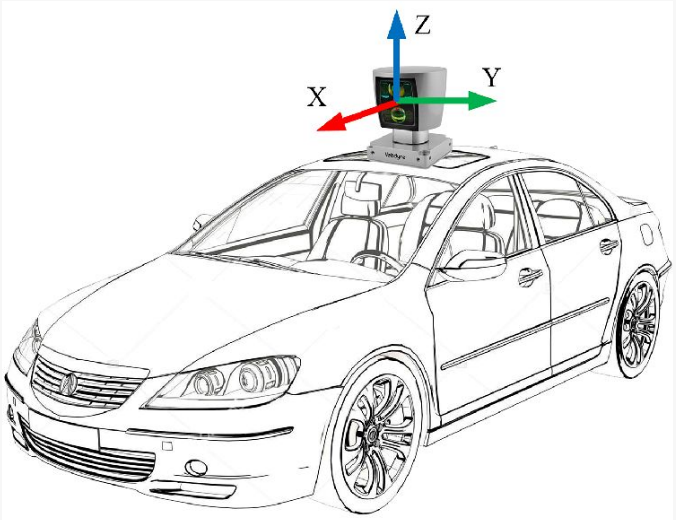


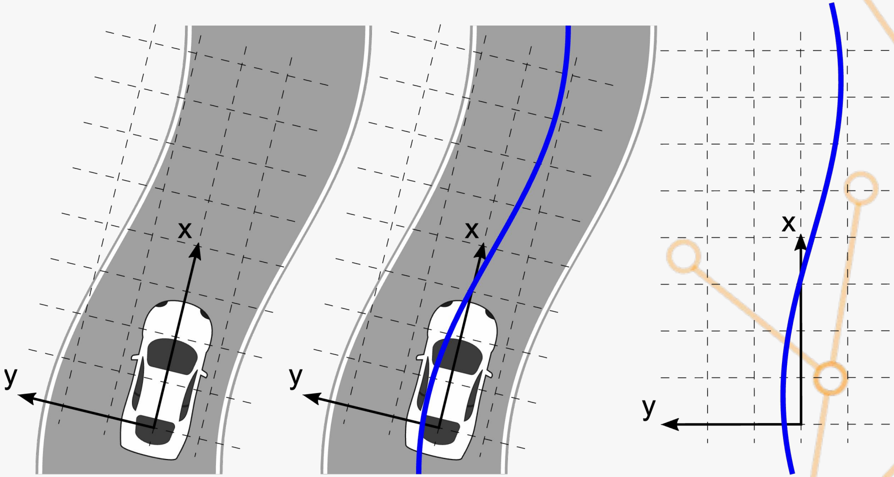
Introduction

Let's make it relatable and motivating

Robots and Self Driving Cars

How do robots or self driving cars know where they are?





Introduction

Let’s make it relatable and motivating

Economics Actuarial Science and Classical ML

Scenario:

An insurance company offers three types of life insurance policies: Term Life, Whole Life, and Universal Life. The company wants to assess the expected payouts for these policies over the next three years based on the probability of death for each age group.

Policy Data (A): This matrix represents the number of policyholders in each age group for each type of policy.

Age Group	Term Life	Whole Life	Universal Life
20-30	1000	500	200
31-40	800	600	300
41-50	500	700	400

Probability Data (B): This matrix represents the probability of death for each age group over the next three years.

Year	20-30	31-40	41-50
1	0.001	0.002	0.004
2	0.0012	0.0025	0.0045
3	0.0013	0.0027	0.005

What if the data has 500,000 rows and 100 columns?

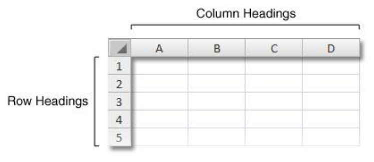
Off Topic

Importance in Various Fields:

Do you know this Lady?

What's her name?

Vectors

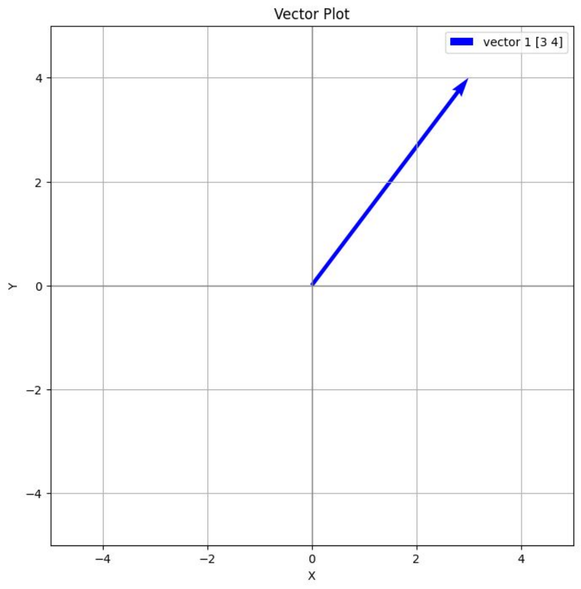


A vector is a mathematical object that has both a magnitude (or length) and a direction. Vectors are often represented as arrows with a certain direction and magnitude or as an ordered list of numbers called components.

Example:

A vector representing a movement of 3 units to the right and 4 units up can be represented as:

$$\vec{v} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$



Types:

Row Vectors:

A $1 \ x \ n$ matrix. (represented horizontally)

Example: $\vec{r} = [3, 4]$; what is the dimension of the vector?

Column Vectors:

An $n \ x \ 1$ matrix. (represented vertically)

Example: $\vec{c} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$; what is the dimension of the vector?

Answer - (1 x 2) and (2 x 1)

Vectors

Dimensions and Coordinates:

Dimensions refer to the number of coordinates needed to specify any point within it. A coordinate is a set of values that provides an exact position within a dimensional space. It's a means to describe the location of a point in various spaces, such as on a line, plane, or in space.

- **1D - Line:** Has length but no width or height.
- **2D - Plane:** Has length and width but no height. Examples: Square, circle, triangle.
- **3D - Volume:** Has length, width, and height. Examples: Cube, sphere, pyramid.
- **4D and Beyond:** We can't visualize 🤔

Operations:

Similar to math where we have operations like addition (+), subtraction (-), multiplication (x); it turns out that vectors can also operate on each other.

The follow some rules as outlined below.

1. Vector addition and dot product is commutative -> $\vec{a} + \vec{b} = \vec{b} + \vec{a}$ and $\vec{a} \cdot \vec{b} = \vec{b} \cdot \vec{a}$
2. Vector addition is associative -> $\vec{a} + (\vec{b} + \vec{c}) = (\vec{a} + \vec{b}) + \vec{c}$
3. Vector dot product is distributive -> $\vec{a} \cdot (\vec{b} + \vec{c}) = (\vec{a} \cdot \vec{b}) + (\vec{a} \cdot \vec{c})$

Vectors

Operations Example:

Addition: Combining two vectors to produce a third vector.

Example: $\begin{bmatrix}3\\4\end{bmatrix}+\begin{bmatrix}1\\2\end{bmatrix}=\begin{bmatrix}4\\6\end{bmatrix}$

Subtraction: Finding the difference between two vectors.

Example: $\begin{bmatrix}3\\4\end{bmatrix}-\begin{bmatrix}1\\2\end{bmatrix}=\begin{bmatrix}2\\2\end{bmatrix}$

Scalar Multiplication: Multiplying a vector by a scalar (a single number).

Example: $2\times\begin{bmatrix}3\\4\end{bmatrix}=\begin{bmatrix}6\\8\end{bmatrix}$

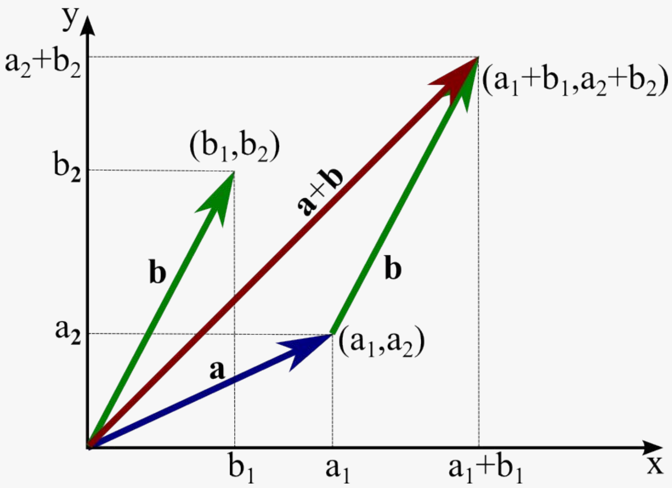
Scalar

Single number - In contrast to other objects in linear algebra, which are usually arrays of numbers.

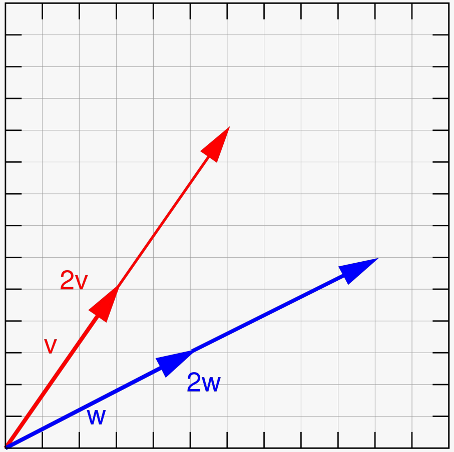
- What does these operations look like graphically?
- What if we have more than 2 rows?
- Lets say 3 row, whats the dimension of the vector?
- What if its 67 rows?

Lets go to colab – https://tinyurl.com/ai6vecmat

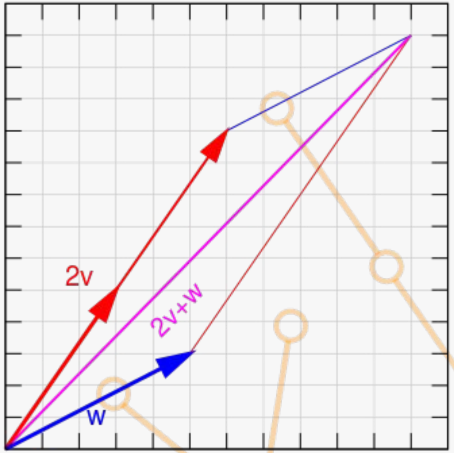
Vectors



Addition



Scaling



???

Vectors

Operations Example:

Addition: Combining two vectors to produce a third vector.

Example: $\begin{bmatrix} 3 \\ 9 \\ -4 \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \\ 7 \end{bmatrix} = ?$

Subtraction: Finding the difference between two vectors.

Example: $\begin{bmatrix} 3 \\ 9 \\ -4 \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \\ 7 \end{bmatrix} = ?$

Scalar Multiplication: Multiplying a vector by a scalar (a single number).

Example: $2 \times \begin{bmatrix} 3 \\ 9 \\ -4 \end{bmatrix} = ?$

Scalar

Single number - In contrast to other objects in linear algebra, which are usually arrays of numbers.

Vectors

Dot Product and Cross Product:

Dot Product: Gives a scalar result. It's the sum of the products of the corresponding entries of the two vectors.

Example:

For vectors $\vec{a} = \begin{bmatrix}1\\2\end{bmatrix}$ and $\vec{b} = \begin{bmatrix}3\\4\end{bmatrix}$

Dot product = $1\times3 + 2\times4 = 11$. Notice that the output of the dot product results to a scalar.

Something to note:

- Idea of dot products don't make sense without understanding the concept of transpose.

- So what is transpose?

- ***Transpose:*** The transpose of a matrix is obtained by swapping its rows with its columns. If the vector/matrix A has dimensions $m\times n$, its transpose, denoted A^T, will have dimensions $n\times m$.

- More formally, if A is a matrix and a_{ij} is an element in the i^{th} row and j^{th} column, then in the transpose A^T, the element a_{ij} will be in the j^{th} row and i^{th} column.

$$\begin{bmatrix}1 & 2 & 3\end{bmatrix} \rightarrow \begin{bmatrix}1\\2\\3\end{bmatrix} \text{ or } \begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} \rightarrow \begin{bmatrix}1 & 4\\2 & 5\\3 & 6\end{bmatrix}$$

Vectors

Dot Product and Cross Product (Again):

Dot Product: Gives a scalar result. It's the sum of the products of the corresponding entries of the two vectors.

Example:

For vectors $\vec{a} = \begin{bmatrix}1\\2\end{bmatrix}$ and $\vec{b} = \begin{bmatrix}3\\4\end{bmatrix}$

Dot product = $\vec{a}^T \cdot \vec{b}$ --> $[1, 2] \cdot \begin{bmatrix}3\\4\end{bmatrix}$ --> $1{\times}3 + 2{\times}4 = 11.$

i.e $(1 \times m) \cdot (n \times 1) \ = (1)\ where\ m \ = \ n$

Cross Product: Gives a vector result. It's perpendicular to the plane determined by the two original vectors. Applicable to 3 coordinate vectors on a 3D space.

Example: For vectors $\vec{a} = \begin{bmatrix}a_x\\a_y\\a_z\end{bmatrix}$ and $\vec{b} = \begin{bmatrix}b_x\\b_y\\b_z\end{bmatrix}$,

The cross-product $\vec{c}$ of vectors $\vec{a}$ and $\vec{b}$ is calculated using the following formula:

$$\vec{c} = \vec{a} \times \vec{b}$$

Vectors

Dot Product and Cross Product:

Cross Product:

Where the components of $\vec{c}$ are given by

$$c_x = a_y \,.\, b_z \ - a_z \,.\, b_y$$
$$c_y = a_z \,.\, b_x \ - a_x \,.\, b_z$$
$$c_z = a_x \,.\, b_y \ - a_y \,.\, b_x$$

For vectors $\vec{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ and $\vec{b} = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$,

Cross product = $\begin{bmatrix} -3 \\ 6 \\ -3 \end{bmatrix}$. [Head over to colab and plot]

Matrices

- A matrix is a rectangular array of numbers, symbols, or expressions, arranged in rows and columns. It's used to represent linear transformations, systems of linear equations, and more.

Example:

A 2x2 matrix: $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \dashrightarrow \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

Rectangular Matrix: Number of rows ≠ Number of columns.

Example: $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

Diagonal Matrix: Non-diagonal elements are zero.

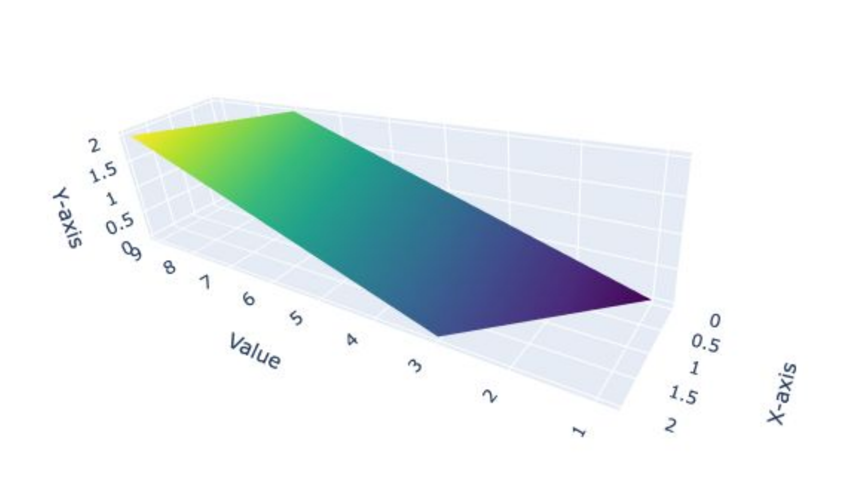
Example: $\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$

Identity Matrix: Diagonal elements are one, others are zero.

Example: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Zero Matrix: All elements are zero.

Example: $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$



Matrices

Operations on Matrices:

Similar to vectors, matrices also have rules.

1. Distributivity over addition: $A(B+C) = AB + AC$
2. Associativity: $A(BC) = (AB)C$
3. Not commutative: $AB = BA$ is not always true
4. Transpose of a matrix product has a simple form: $(AB)^T = B^T A^T$

Addition: Element-wise addition of two matrices of the same size (should be of same size).

Example: $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 5 & 7 \\ 9 & 11 \end{bmatrix}$

Subtraction: Element-wise subtraction of two matrices of the same size (should be of same size).

Example: $\begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}$

Multiplication: Dot product of rows of the first matrix with columns of the second matrix.

Example: $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 16 & 19 \\ 36 & 43 \end{bmatrix}$??? What happened here ???

Matrices

Operations on Matrices:

Remember dot products in vectors? Well, it's the same concept.

Example: $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 1\times4+2\times6 & 1\times5+2\times7 \\ 3\times4+4\times6 & 3\times5+4\times7 \end{bmatrix} = \begin{bmatrix} 16 & 19 \\ 36 & 43 \end{bmatrix}$

What if we have this kind of matrix

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix} = ?$

Is this valid?

It will not work be because the condition for a valid dot product is $(m\times n)\cdot(n\times p) = (m\times p)$

This means we should transpose the matrix on the right. (Do you remember transpose?)

$\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \times \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix} =$? Calculate this

Matrices

Determinant and Inverse:

Determinant: A scalar value derived from a square matrix. It provides information about the matrix's properties.

Example (for 2x2 matrix): For $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$, Determinant (Det) = $ad - bc$,

Inverse:

For a matrix A, its inverse $A^{-1}$is a matrix such $A{\times}A^{-1} = I$, where I is the identity matrix.

- Identity matrix does not change value of vector when we multiply the vector by identity matrix.

- How can we calculate the inverse of a matrix?

$$A^{-1} = \frac{1}{Det}\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Example:

For $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, Inverse = $\begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix}$. Calculate with hand or code.

Matrices

Linear Transformations:

A linear transformation, often denoted as T, is a function from one vector space to another that preserves the operations of vector addition and scalar multiplication. In simpler terms, it's a way to "move" vectors around without changing the essence of the space's structure.

PUT SIMPLY -A way to find unknowns with matrices.

$$Ax = b$$

$$A = \begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

PUT SIMPLY - Sometimes we wish to solve for the unknowns $x = \{x_1, .., x_n\}$ when A and b provide constraints.

What if we want to find x?

$$Ax = b$$
$$A^{-1}\, Ax = A^{-1}b$$
$$I \cdot x = A^{-1}b$$
$$x = A^{-1}b$$

Matrices

Linear Transformations:

How is this useful to me?

Simultaneous Equations:

$$2x + 3y = 12 \quad \dots(1)$$
$$4x - 2y = 10 \quad \dots(2)$$

$$\begin{bmatrix} 2 & 3 \\ 4 & -2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \end{bmatrix} \quad \dots\dots\dots. \quad Ax = b$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.1 & 0.3 \\ 0.4 & -0.2 \end{bmatrix} \cdot \begin{bmatrix} 12 \\ 10 \end{bmatrix}. \quad \dots\dots\dots \quad x = A^{-1}b$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4.2 \\ 2.8 \end{bmatrix}$$

What if we have 150 unknowns (x, y, z, Ɛ, e, α, β ...), leading to 150 equations?
Lets go to colab.

Sparse Matrices

Definition and Importance: A sparse matrix is a matrix in which most of the elements are zero (or the default value). Storing sparse matrices efficiently is crucial because it saves memory and computational resources.

Why it's Important:

Memory Efficiency: Reduces the storage space required.

Computational Efficiency: Faster matrix operations by skipping zero elements.

Applications:

1. **Large Datasets:** Representing data with a significant number of missing or default values.
2. **Graph Algorithms:** Adjacency matrices of large graphs where most nodes are not directly connected.
3. **Finite Element Methods:** In engineering simulations where the system matrix is often sparse.
4. **Natural Language Processing:** Representing word-document matrices in text mining tasks.
5. **Machine Learning:** Storing large data sets with many irrelevant features.

Sparse Matrices

Storage Techniques:

Compressed Sparse Row (CSR):

- Also known as Row Compressed Form.

- Three one-dimensional arrays are used: one for non-zero values, one for column indices of elements, and one for the starting index of the first non-zero element in each row.

Example: For matrix

0	1	0	2
1	0	3	0
0	0	0	0
4	0	5	0

CSR representation:

None-Zero Values - the non-zero values - [1, 2, 1, 3, 4, 5]

Column Indices - the column indices of these non-zero values - [1, 3, 0, 2, 0, 2]

Row Pointers - For each row, add the number of non-zero values in that row to the previous row pointer value. [0, 2, 4, 4, 6]

Two non-zero values (so add 2 to the previous pointer which is 0)
Row 2: Two non-zero values (so add 2 to the previous pointer which is 2)
Row 3: Zero non-zero values (so add 0 to the previous pointer which is 4)
Row 4: Two non-zero values (so add 2 to the previous pointer which is 4)

Sparse Matrices

Storage Techniques:

Compressed Sparse Row (CSR):

CSR representation:

None-Zero Values - [1, 2, 1, 3, 4, 5]

Column Indices - [1, 3, 0, 2, 0, 2]

Row Pointers - [0, 2, 4, 4, 6]

CSR Reverse:

1. Start with an empty matrix filled with zeros.

2. For the first row, the range of values is from index 0 to 2 (exclusive) in the Values array. So, the values are [1, 2], and their column indices are [1, 3]. Place these values in the first row at columns 1 and 3.

3. For the second row, the range of values is from index 2 to 4 (exclusive). So, the values are [1, 3], and their column indices are [0, 2]. Place these values in the second row at columns 0 and 2.

4. The third row has no non-zero values (as indicated by the repeated value in the Row Pointers array).

5. For the fourth row, the range of values is from index 4 to 6 (exclusive). So, the values are [4, 5], and their column indices are [0, 2]. Place these values in the fourth row at columns 0 and 2.

Sparse Matrices

Storage Techniques:

Compressed Sparse Column (CSC):

- Also known as Column Compressed Form.

- Similar to CSR but compresses the column indices.

Example: For matrix

$$\begin{bmatrix} 0 & 1 & 0 & 2 \\ 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 4 & 0 & 5 & 0 \end{bmatrix}$$

CSC representation:

None-Zero Values - the non-zero values column-wise - [1, 4, 1, 3, 5, 2]

Row Indices - the row indices of these non-zero values - [1, 3, 0, 1, 3, 0]

Column Pointers- For each column, add the number of non-zero values in that column to the previous column pointer value - [0, 2, 3, 5, 6]

Lets take a brief look at images

Let look and Ladi's picture again. --> Colab

Check the shape of the colored image.

Can we manipulate the picture?

Recommended material to more study

1. 3Blue1Brown - https://www.youtube.com/@3blue1brown/
2. Khan Academy – Linear Algebra
3. MITOPENCOURSEWARE – Linear Algebra - Prof. Gilbert Strang

Thank You