**Intro to Linear Algebra**

1. Data Structures for Algebra
2. Common Tensor Operations
3. Matrix Properties

* What Linear Algebra is?
* A brief history of algebra (Al-Khwarizmi)
* Tensors
* Scalars
* Vectors and Vector Transposition
* Norms and Unit vectors
* Basis, Orthogonal and Orthonormal vectors
* Arrays in Numpy
* Matrices
* Tensors in TensorFlow and PyTorch

Linear algebra is “solving for unknowns within system of linear equations”.

One solutions, no solutions or infinite solutions.

Brief history of Algebra consists of:

Babylonians (1900 BC), Egyptians, Indians, Greeks, Chinese (250 BC), Arabic to Latin translation (12th century), rivaled others (13th century).

**Contemporary applications:**

* Solving for unknowns in ML algos, including deep learning
* Reducing dimensionality (eg. Principal component analysis)
* Ranking results (eg. with eigenvector)
* Recommenders (eg. singular value decomposition, SVD)

**Tensors**

ML generalization of vectors and matrices to any number of dimensions.

Scalar – x (magnitude only)

Vector – [x1, x2, x3] (array)

Matrix – [x11, x12] (flat table, square)

[x21, x22]

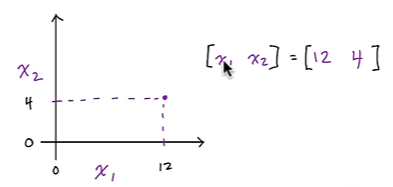
3 – Tensor –

(3d table,

cube)

**Vectors**

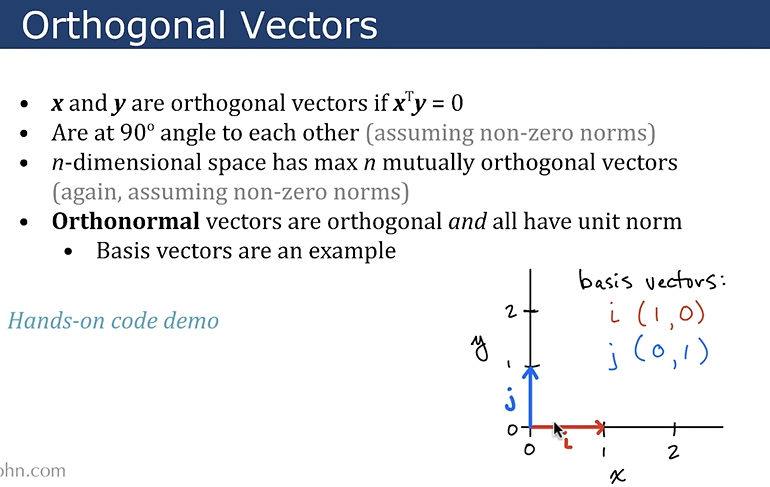
* One-dimensional array of numbers
* Denoted in lowercase, italic, bold
* Arranged in an order so element can be accessed by its index
* Representing a point in space:



**The transpose of row vector is a column vector.**

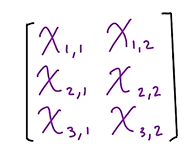
What Is the L² Norm / Euclidean Norm? L² is the most commonly used norm and the one most encountered in real life. The L² norm measures the shortest distance from the origin. It is defined as the root of the sum of the squares of the components of the vector.

Also known as Manhattan Distance or Taxicab norm. L1 Norm is the sum of the magnitudes of the vectors in a space. It is the most natural way of measure distance between vectors, that is the sum of absolute difference of the components of the vectors.



**Matrices**

* Two-dimensional array of numbers
* Denoted in uppercase, italics, bold: ***X***
* Height given priority ahead of width in notation: (nrow, ncol)
* Individual scalar elements denoted in uppercase, italics only: *X*1,2
* Colon represents an entire row or column:



We use nested brackets for creating matrices in Numpy:

X = np.array([[25, 2], [5, 26], [3, 7]])

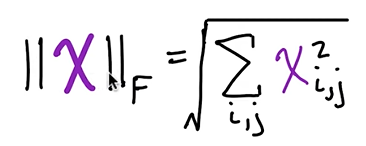
If two tensors have the same size, operations are often by default applied element-wise. This is **not matrix multiplication**, which we'll cover later, but is rather called the **Hadamard product** or simply the **element-wise product**.

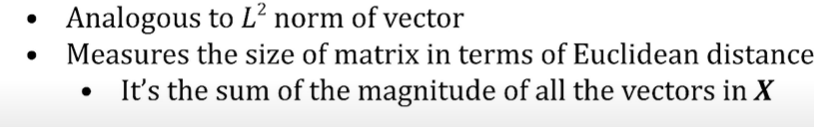
The mathematical notation is 𝐴⊙𝑋

It is not a matrix multiplication, operation in here goes like this:

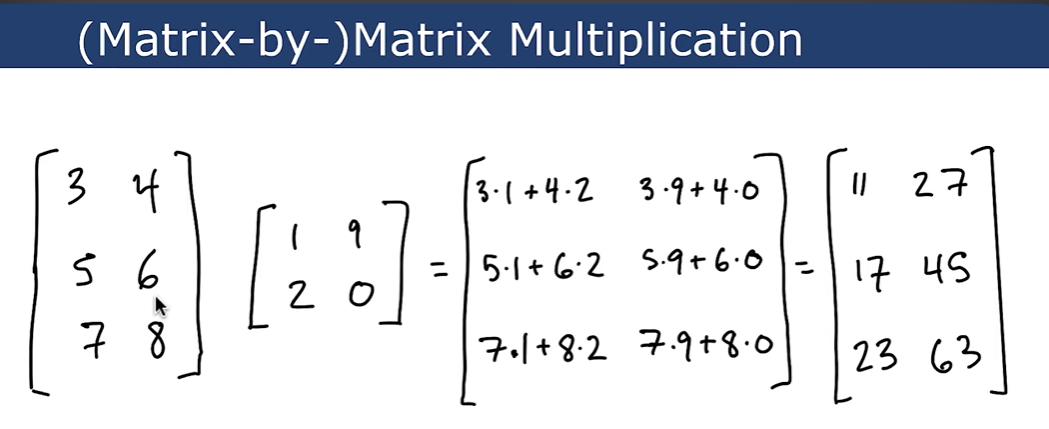
Imagine we have 2matrices, X and Y. To find Hadamard product, X11\*Y11; X12\*Y12 and etc.

**Frobenius Norm:**





Np.linalg.norm(X)



**Some Notes:**

If ***X*** is a *symmetric matrix*,

* It must be square
* Transpose of it equals itself: ***XT = X***

If ***I*** is an identity matrix, (it is a special kind of diagonal matrix)

* Every element along main diagonal is 1
* All other elements are 0
* Notation: ***In*** where ***n = height (or width)***

**INVERSION OF MATRICES**

Matrix inverse of ***X*** is denoted as ***X***-1

***X***-1 \* ***X = I***n

**Linear Algebra 2**

**Eigenvalues and eigenvectors:**

Let’s say we have a painting of Mona Lisa and to understand what eigenvector is we have drawn a grid top of the Mona Lisa. And we have basis vectors here, namely orthogonal vectors (red-vertical, blue-horizontal). If we apply a flipping matrix on the painting, it reflects Mona Lisa over the ***y*** axis. Red and blue vectors both are eigenvectors, because both of the vectors retain their direction (blue is just negative but it is still on its span).

If we apply a shearing matrix to our image, the red vector is no longer an eigenvector. Pixels have moved in here right on top and left on bottom.

Eigenvalues are scalar values that tell you how much eigenvector length has changed as a result of applying the particular matrix (in this case it is a shearing matrix). In our image example, after applying matrix, the eigenvalue of blue eigenvector is 1 (eigenvalue = 1). **Note that**, if Mona Lisa gets widened or narrowed, it means the eigenvalue increases or decreases, respectively.

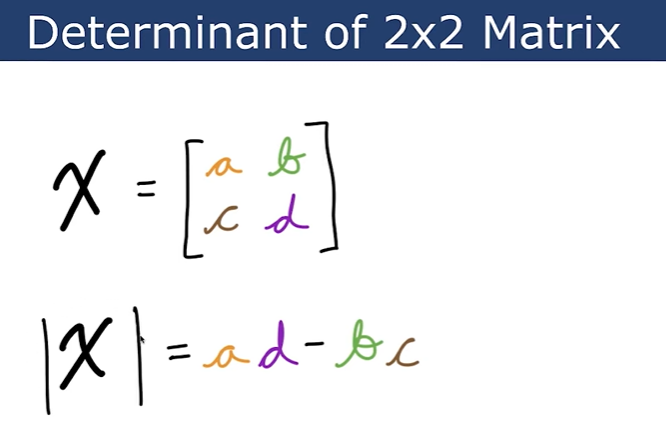
Eigenvalue can also be negative, e.g., a new shearing-and-flipping matrix has the same eigenvector as shearing-only matrix but its eigenvalue = -1. If eigenvector were to double in length while reversing direction, eigenvalue would be -2.

**eig()** method helps us to use eigenvector and eigenvalues in NumPy, which returns tuples:

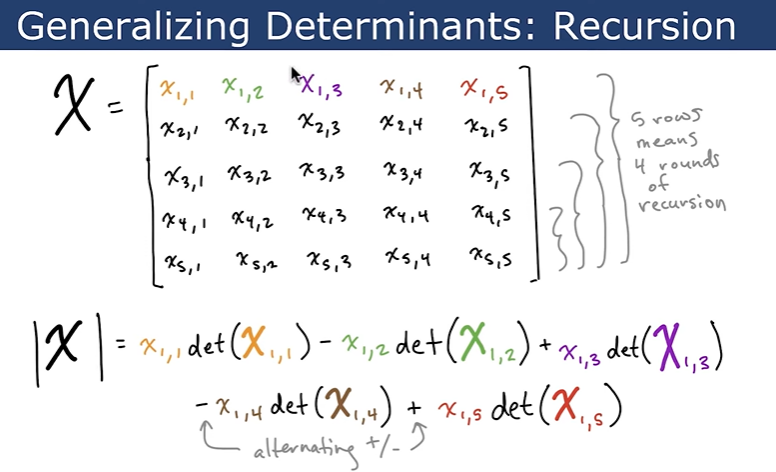
* A vector of eigenvalues
* A matrix of eigenvectors

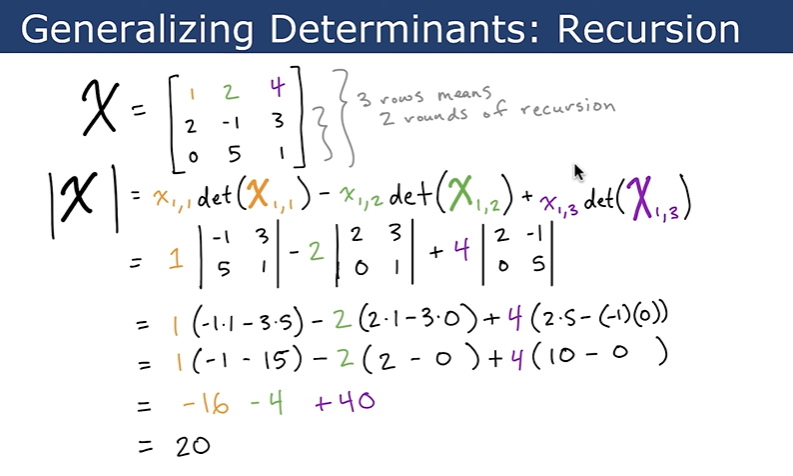
**DETERMINANTS**

Det() method can solve determinants in NumPy



Finding determinants of 5\*5 matrices:

****



**Note: Determinant of a matrix is product of all eigenvalues of *X***

***Det(X) =* product of all eigenvalues of *X***

In Numpy, we can use product() function to solve this.

**EIGENDECOMPOSITION**



The decomposition of a matrix into eigenvectors and eigenvalues reveals characteristics of the matrix, e.g:

* Matrix is singular bcs its eigenvalues are zero

**PRINCIPAL COMPONENT ANALYSIS**

* Simple Machine learning algorithm
* Unsupervised: enables identification of structure in unlabeled data (e.g. we can have some flowers but don’t know their labels)
* Involves many linear algebra concepts:
  + Norms
  + Orthogonal and identity matrices
  + Trace operators

**RECOMMENDED FOR FURTHER STUDY**

* Basic Algebra
  + Khan academy
  + 3Blue1Brown on YouTube
* Linear Algebra
  + 3Blue1Brown on YouTube
  + Ch.2 of Goodfellow et al. 2016 – Deep Learning
  + Ch.2 of Deisenroth et. Al 2020 – Mathematics for ML
  + Sheldon Axler’s 2015 – Linear Algebra done right
* Next steps in ML Foundation series
  + - Calculus 1: Limits and derivatives
    - Calculus 2: Partial derivatives & Integrals
    - Probability & Information Theory
    - Optimization