Mathematics

# Linear Algebra for Data Science

## Matrix

1. Scalar: A scalar is a single quantity or value that is fully described by its magnitude (size) and can be represented as a real number. Scalars have no direction and are commonly used to represent quantities like temperature, mass, time, or distance. For example, the number 5 is a scalar.
2. Vector: A vector is a quantity that has both magnitude and direction. It is represented by an ordered set of numbers, typically written as a column or row of values enclosed in brackets. Vectors are used to represent physical quantities with direction, such as displacement, velocity, or force.

For instance, [3, -2, 7] is a 3-dimensional vector.

In Python:

V = np.array([4,5,6])

1. Matrix: A matrix is a rectangular array of numbers (or elements) arranged in rows and columns. It is represented by enclosing the elements in double brackets. Matrices are used for various mathematical and engineering operations, including linear transformations, solving systems of equations, and computer graphics transformations.

An example of a 2x3 matrix is:

[ [ 1 2 3 ]

[ 4 5 6 ] ]

In Python:

V = np.array([[1,2,3],[4,5,6]])

In summary:

* Scalars are single numerical values with no direction.
* Vectors are quantities with magnitude and direction, represented as ordered sets of numbers.
* Matrices are rectangular arrays of numbers used for mathematical operations.

### Tensors

A tensor is an array of numbers or functions that transform under certain rules when changing the coordinate system. These rules ensure that tensors have coordinate-independent meaning and can be used consistently in different coordinate systems.

Tensors are characterized by their rank, which corresponds to the number of indices needed to specify each component. Here are some common ranks of tensors:

1. Rank-0 tensor (Scalar): A scalar is a tensor of rank 0, representing a single value, just like the scalar we discussed earlier.
2. Rank-1 tensor (Vector): A vector is a tensor of rank 1, representing a list of numbers with both magnitude and direction.
3. Rank-2 tensor (Matrix): A matrix is a tensor of rank 2, representing a rectangular array of numbers.
4. Rank-3 tensor: A tensor of rank 3 requires three indices to specify each component.

And so on, you can have tensors of higher ranks.

Tensors play a crucial role in various branches of physics, including general relativity, electromagnetism, fluid dynamics, and quantum mechanics. In data science, deep learning models often use tensors to represent and process multi-dimensional data, such as images and time series.

### Matrix Operations

1. Matrix Addition and Subtraction: Matrix addition and subtraction are performed element-wise, meaning each corresponding element in the matrices is added or subtracted to produce the result.

For example, let's consider two matrices:

matrix1 = [[1, 2], [3, 4]]

matrix2 = [[5, 6], [7, 8]]

Matrix Addition:

result\_addition = matrix1 + matrix2

The result will be:

result\_addition = [[1+5, 2+6], [3+7, 4+8]] = [[6, 8], [10, 12]]

Matrix Subtraction:

result\_subtraction = matrix1 - matrix2

The result will be:

result\_subtraction = [[1-5, 2-6], [3-7, 4-8]] = [[-4, -4], [-4, -4]]

1. Matrix Multiplication: Matrix multiplication can be performed using two methods: element-wise multiplication (Hadamard product) and matrix multiplication (dot product).

Element-wise Multiplication (Hadamard product):

result\_elementwise\_mult = matrix1 \* matrix2

The result will be:

result\_elementwise\_mult = [[1\*5, 2\*6], [3\*7, 4\*8]] = [[5, 12], [21, 32]]

Matrix Multiplication (Dot product):

result\_dot\_product = np.dot(matrix1, matrix2)

Or using the "@" operator:

result\_dot\_product = matrix1 @ matrix2

The result will be:

result\_dot\_product = [[1\*5 + 2\*7, 1\*6 + 2\*8], [3\*5 + 4\*7, 3\*6 + 4\*8]] = [[19, 22], [43, 50]]

1. Transpose of a Matrix: Transposing a matrix means interchanging its rows and columns. It effectively flips the matrix over its main diagonal.

For example, if we have:

matrix1 = [[1, 2], [3, 4]]

Transposing **matrix1** will result in:

result\_transpose = matrix1.T = [[1, 3], [2, 4]]

1. Inverse of a Matrix: The inverse of a square matrix A is denoted as A^(-1) and has the property that A \* A^(-1) = I (the identity matrix).

For example, consider the square matrix:

matrix1 = [[2, 1], [1, 3]]

The inverse of **matrix1** will be:

result\_inverse = np.linalg.inv(matrix1) = [[3, -1], [-1, 2]]

It is important to note that not all matrices have an inverse. Non-square matrices and singular matrices (determinant = 0) do not have inverses.

1. Determinant of a Matrix: The determinant of a square matrix provides valuable information about its properties and invertibility. For a 2x2 matrix:

matrix1 = [[a, b], [c, d]]

The determinant is given by: det(matrix1) = ad - bc.

1. Solving a System of Linear Equations: In the context of solving a system of linear equations Ax = b, where A is a square matrix and x is the unknown vector, the solution can be found using the inverse of A:

A = [[2, 1], [1, 3]] b = [4, 5]

The solution x can be obtained as follows:

result\_solution = np.linalg.solve(A, b) = [1, 1]

So the solution is x = [1, 1].

These are the explanations for the common matrix operations in Python using NumPy. NumPy's array operations are optimized for efficiency, making it a powerful tool for performing numerical computations involving matrices and arrays.