Scalars, Vectors, and Matrices

Monower Hossen

1. Scalars

1.1 Definition

A **scalar** is a single numerical value belonging to a field, typically the real numbers $\mathbb{R}$ or complex numbers $\mathbb{C}$.

$$\alpha \in \mathbb{R}$$

Scalars represent **magnitude only**, without direction.

1.2 Properties

- Scalars obey standard arithmetic operations: addition, subtraction, multiplication, and division.
- They are invariant under coordinate transformations.
- Scalars can scale vectors and matrices.

1.3 Role in Machine Learning

In machine learning, scalars are used to represent:

- Learning rates
- Bias terms
- Loss values
- Individual feature values

For example, the loss value returned by a model during training is a scalar.

1.4 Scalar Operations

Given,

$$\alpha = 5, \beta = -3.2, \gamma = \sqrt{2}$$

Addition:

$$\alpha + \beta = 5 + (-3) = 2$$

Multiplication:

$$\alpha \cdot \beta = 4 \times 2 = 8$$

Scalar power:

$$\alpha^2 = 3^2 = 9$$

1.5 Role in Machine Learning

- Learning rate (η)
- Regularization coefficient (λ)
- Bias term (b)

2. Vectors

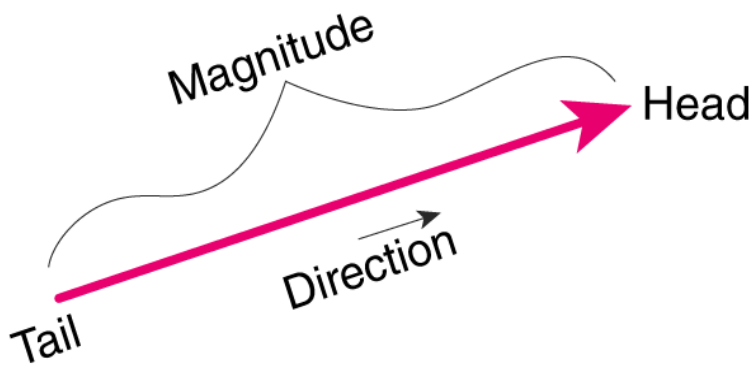
2.1 Definition

A **vector** is an ordered collection of scalars, representing magnitude and direction. $\mathbf{x} \in \mathbb{R}^n$

Vectors are used to represent:

- **Forces** in physics
- **Feature sets** in machine learning
- **Word embeddings** in NLP

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$



2.2 Vector Types

- **Column vector**: $n \times 1$
- **Row vector**: $1 \times n$

2.3 Example

$$\mathbf{x} = \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix}$$

2.4 Vector Operations

(a) Vector Addition

$$\mathbf{x} + \mathbf{y} = \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix} + \begin{bmatrix} 1 \\ 4 \\ -2 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ 1 \end{bmatrix}$$

(b) Scalar Multiplication

$$2\mathbf{x} = 2\begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix} = \begin{bmatrix} 4 \\ -2 \\ 6 \end{bmatrix}$$

(c) Dot Product

$$\mathbf{x} \cdot \mathbf{y} = (2)(1) + (-1)(4) + (3)(-2) = 2 - 4 - 6 = -8$$

(d) Euclidean Norm

$$\| \mathbf{x} \|_2 = \sqrt{2^2 + (-1)^2 + 3^2} = \sqrt{14}$$

2.5 Interpretation

- Vectors represent **data points**, **feature vectors**, and **embeddings**.
- Dot product measures **similarity**.

In Python, we can create a vector, for example, using the NumPy library:

```
import numpy as np

my_vector = np.array([2, 0.2, 0.5, 5, 1])
print(my_vector)
```

Output:

```
[2.  0.2 0.5 5.  1. ]
```

3. Matrices

3.1 Definition

A **matrix** is a 2-dimensional array of scalars arranged in rows and columns.

$$\mathbf{A} \in \mathbb{R}^{m \times n}$$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

3.2 Example

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

3.3 Matrix Operations

(a) Matrix Addition

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$= \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

(b) Matrix Multiplication

$$\mathbf{AB} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$= \begin{bmatrix} (1 \cdot 5 + 2 \cdot 7) & (1 \cdot 6 + 2 \cdot 8) \\ (3 \cdot 5 + 4 \cdot 7) & (3 \cdot 6 + 4 \cdot 8) \end{bmatrix}$$

$$= \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

(c) Matrix–Vector Multiplication

$$\mathbf{Ax} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

3.4 Transpose

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$\mathbf{A}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

In Python, a matrix can be created using NumPy:

```
import numpy as np

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

print(M.shape)
print(M[0, 1]) # Retrieving a specific element of a matrix
```

Output:

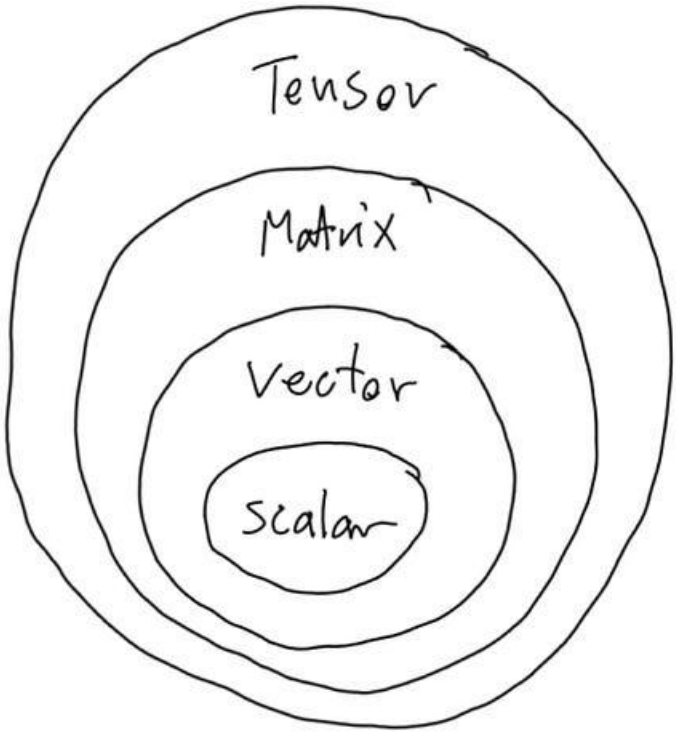
```
(2, 3)
2
```

4. Relationship Between Scalars, Vectors, and Matrices

Object	Dimension	ML Interpretation
Scalar	1×1	Bias, loss value
Vector	$n \times 1$	Feature vector
Matrix	$m \times n$	Dataset, weights

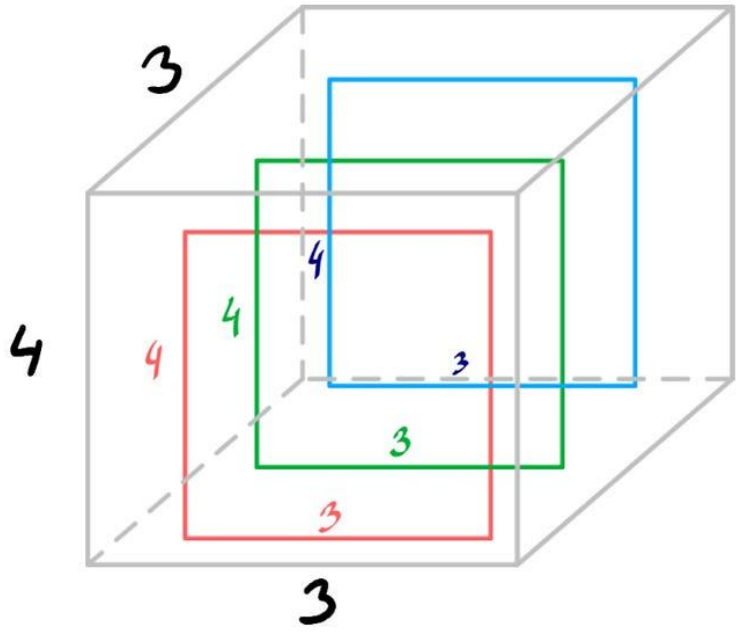
Tensor

A tensor is a mathematical object that can have any number of dimensions. We can think of it as a generalization of concepts like scalar, vector, and matrix.



A tensor can contain all three of these objects within it

The number of dimensions in a tensor is called its rank. A scalar, which we mentioned earlier, has no dimensions and is therefore a 0^{th}-rank tensor. A vector has one dimension, making it a 1^{st}-rank tensor, while a matrix is a 2^{nd}-rank tensor. A 3^{rd}-rank tensor, for example, can represent an RGB image consisting of three matrices – one for each color channel (red, green, and blue).



Example of a 3×4×3 tensor

A tensor can therefore be seen as a „container” for data that can hold any of the objects mentioned above.

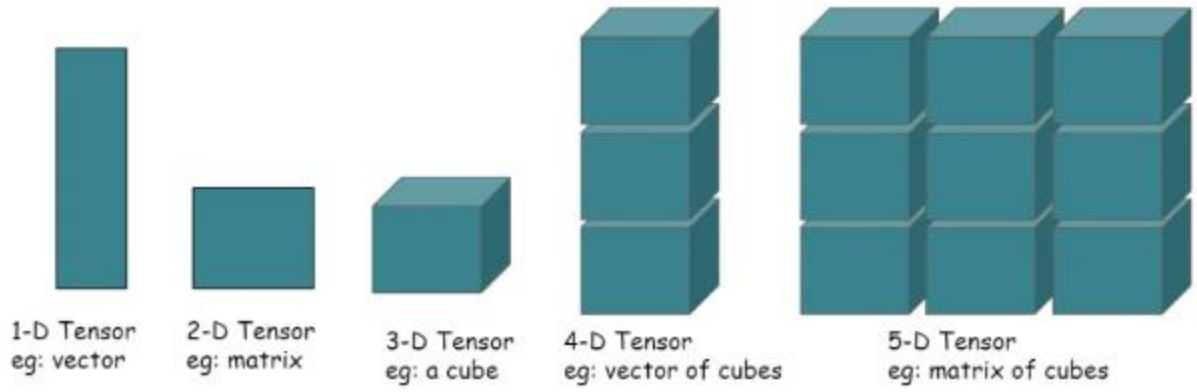
Example of a 3D tensor in Python using the NumPy library:

```
import numpy as np

tensor = np.random.rand(3, 4, 3)
print(tensor)
print(tensor.shape)
```

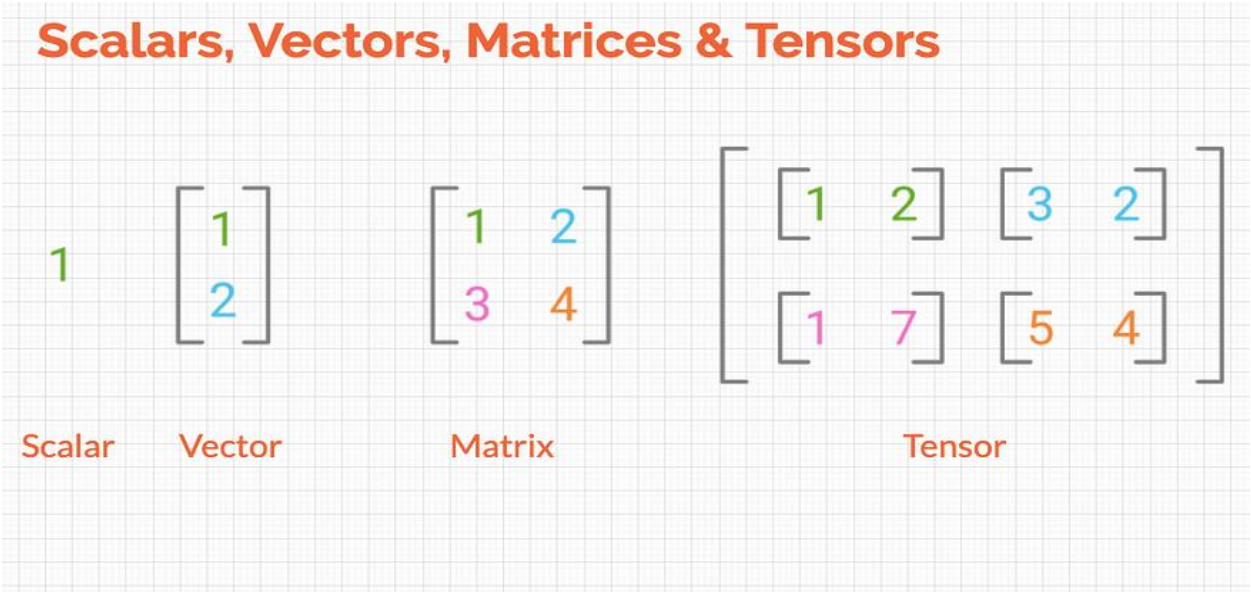
Outputs:

```
[[[0.17187468  0.75221816  0.72999418]
  [0.87890064  0.12073509  0.71086969]
  [0.32036432  0.2754748   0.11855517]
  [0.55047325  0.16468913  0.81651425]]

 [[0.54240796  0.89802446  0.45886324]
  [0.39255608  0.92778486  0.13720558]
  [0.77867887  0.44295862  0.21003297]
  [0.52421613  0.29731313  0.05817399]]

 [[0.60105893  0.87168155  0.9977958 ]
  [0.18845311  0.7383102   0.80534301]
  [0.12040446  0.57731904  0.6915367 ]
  [0.72561021  0.19891586  0.69639787]]]
(3, 4, 3)
```

Here, 3 represents the number of tensor objects (depth), 4 is the height, and 3 is the width of the inner objects.



Tensors with different dimensions



6. Application in Machine Learning (Linear Model)

$$\hat{y} = \mathbf{w}^T\mathbf{x} + b$$

Where:

- $\mathbf{w}$→ weight vector
- $\mathbf{x}$→ input vector
- b→ scalar bias

This equation combines **scalar + vector + matrix concepts** into a single predictive model.

Example of Python using the NumPy library:

```
import numpy as np
# Scalar
S = 10

# Vector
V = np.array([1, 2])

# Matrix
M = np.array([[1, 2, 3], [4, 5, 6]])

# Tensor
T = np.array([[[1, 2, 3], [4, 5, 6], [7, 8, 9]],
              [[11, 12, 13], [14, 15, 16], [17, 18, 19]],
              [[21, 22, 23], [24, 25, 26], [27, 28, 29]],
             ])
```