Linear Algebra-Scalars, Vectors, Matrices and Tensors

Scalar

24

Vector

$\begin{bmatrix} 2 & -8 & 7 \end{bmatrix}$

row

or column

$\begin{bmatrix} 2 \\ -8 \\ 7 \end{bmatrix}$

Matrix

$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}$

row(s) × column(s)

Preliminaries

Before start to learn into deep learning

(i) Techniques for storing and manipulating data;

(ii) Libraries for ingesting and preprocessing data from a variety of sources;

(iii) knowledge of the basic linear algebraic operations that we apply to high-dimensional data elements;

(iv) Just enough calculus to determine which direction to adjust each parameter in order to decrease the loss function;

(v) The ability to automatically compute derivatives so that you can forget much of the calculus you just learned;

(vi) Some basic fluency in probability, our primary language for reasoning under uncertainty; and

(vii)Some aptitude for finding answers in the official documentation when you get stuck.

Data Manipulation

There are two important things we need to do with data:

(i) Acquire them;

(ii) Process them once they are inside the computer

- There is no point in acquiring data without some way to store it, so to start, let's get our hands dirty with *n*-dimensional arrays, which we also call *tensors*.

- Datasets into tensors and manipulate these tensors with basic mathematical operations.
- In machine learning the majority of data is most often represented as vectors, matrices, or tensors. Therefore ML heavily relies on Linear Algebra.

Linear Algebra

- Linear algebra is the branch of mathematics that focuses on linear equations.
- It is often applied to the science and engineering fields, specifically machine learning.
- Linear algebra is also central to almost all areas of mathematics like geometry and functional analysis.
- Data is represented by linear equations, which are presented in the form of matrices and vectors.
- Operations on the image, such as cropping, scaling, shearing, and so on are all described using the notation and operations of linear algebra.

One Hot Encoding

- One hot encoding is a process of converting categorical data variables so they can be provided to machine learning algorithms to improve predictions.
- Categorical data refers to variables that are made up of label values, for example, a “color” variable could have the values “red,” “blue,” and “green.” Think of values like different categories that sometimes have a natural ordering to them.
- Some machine learning algorithms can work directly with categorical data depending on implementation, such as a decision tree, but most require any inputs or outputs variables to be a number, or numeric in value.
- This means that any categorical data must be mapped to integers.
- One hot encoding is one method of converting data to prepare it for an algorithm and get a better prediction.

One Hot Encoding

Example for One Hot Encoding

- Convert each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns.
- Each integer value is represented as a binary vector.
- All the values are zero, and the index is marked with a 1.

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Mathematical Objects

- Scalar - 35 (0D Tensor)
- Vector – [6,-8, 9] Row, $\begin{bmatrix} 2, \\ 6, \\ 9 \end{bmatrix}$ Column (1D Tensor)
- Matrix : **Matrix:** $\begin{bmatrix} 2, & -6, & 9 \\ 4, & 5, & -7 \end{bmatrix}$ row(s) x column(s) (2D Tensor)

Tensor

- A tensor is a container for numerical data. It is the way we store the information that we'll use within our system.
- Three primary attributes define a tensor:

  Rank

  Shape

  Data type

- These three indices, where the first one points to the row, the second to the column, and the third one to the axis.

Scalar

- Scalars are just numbers. They have magnitude but no direction.
- Scalars are commonly used to denote quantities such as temperature, time, mass, or distance.
- Any single value from our dataset would represent a scalar.
- The number of bedrooms in our house price data, for example, would be a scalar.
- If we only used one feature as an input from our house price data, then we could represent that as a scalar value

```
1 house_price_df.iloc[[0]]
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built
0	7129300520	20141013T000000	221900.0	3	1.0	1180	5650	1.0	0	0	...	7	1180	0	1955

```
print(house_price_df['bedrooms'].values[0])
3
```

Scalar Cont'd

Learning Rate:

- The learning rate is a scalar value that determines the step size during the optimization process of training a deep learning model.
- It controls how quickly or slowly the model adapts to the training data.
- Adjusting the learning rate influences the convergence and performance of the model.

Activation Functions:

- Activation functions, such as the sigmoid or ReLU, take scalar inputs and apply non-linear transformations to introduce non-linearity in neural networks.
- These scalar-valued functions play a critical role in enabling the network to learn complex patterns and relationships in the data.

Scalar Cont'd

Bias Term:

- In deep learning, a bias term is often included in each layer of a neural network.
- This scalar value is added to the weighted sum of inputs and activation functions, allowing the network to learn offset or shift from the origin.

Confidence Scores:

- In classification tasks, deep learning models produce scalar-valued confidence scores or probabilities to indicate the model's confidence in its predictions.
- These scalar values reflect the model's belief in the predicted class and play a crucial role in decision-making and evaluating the model's performance.

Vectors

- A vector is a mathematical object that represents a collection of values or features.
- It is an ordered list of numbers
- Each number corresponds to a specific component or dimension of the vector.
- Vectors in deep learning are typically represented as column vectors, meaning they are written vertically.
- A vector in deep learning is denoted as $x = [x_1, x_2, x_3, \ldots, x_n]^T$
- Here, x represents the vector, and x_1, x_2, x_3, ..., x_n are the individual components of the vector.
- The superscript ^{T} indicates the transpose operation, which converts a row vector into a column vector.

Vectors Cont'd

Example

- There seems to be more than one usable feature in our house price data.
- How would we represent multiple features?
- The total square footage of the house would be a useful piece of information to have when trying to predict a house price.
- In its most simple format, we can think of a vector as a 1-D data structure

```
# This is a numpy row vector
print(house_price_df[["bedrooms", "sqft_lot"]].values[:1])
[[   3 5650]]
```

- We can also create a 1-D vector by arranging the data in a column, rather than row, format:

```
# This is a numpy column vector
print(house_price_df[["bedrooms", "sqft_lot"]].values.T[:2, :1].shape)
[[   3]
 [5650]]
```

Vector operations

- Element-wise multiplication involves multiplying each corresponding element of two vectors together.
- This operation is commonly used in various deep learning applications, such as attention mechanisms, weighted computations, and element-wise interactions.
- <u>Example 1:</u>
- Vector A: [2, 3, 4]
- Vector B: [1, 2, 3]
- Element-wise multiplication, $A \odot B = [2*1, 3*2, 4*3] = [2, 6, 12]$
- <u>Example 2:</u>
- Vector X: [1, 2, 3]
- Vector Y: [4, 5, 6]
- Dot product, $X \cdot Y = (1*4) + (2*5) + (3*6) = 4 + 10 + 18 = 32$

Matrices

- Matrices play a significant role in deep learning as they provide a structured way to represent and manipulate data.
- In deep learning, matrices are used to represent various components such as input data, weights, activations, and gradients in neural networks.
- They enable efficient computation and manipulation of data within neural networks, supporting various applications across image processing, natural language processing, recommender systems, and generative modeling

Matrices Cont'd

Examples:

Image Processing: Used to represent images in deep learning for tasks like image classification, object detection, and image generation.

Natural Language Processing: Matrices are employed to represent textual data in tasks such as sentiment analysis, language translation, and text generation.

Recommender Systems: Matrices are utilized in collaborative filtering-based recommendation systems to represent user-item interactions and calculate similarity or preference scores.

Generative Models: Matrices are used in generative models like Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) to generate new samples by manipulating latent space vectors.

Matrix operations

- Element-wise Operations: Element-wise operations are performed on corresponding elements of two matrices or a matrix and a scalar.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} a+1 & b+2 \\ c+3 & d+4 \end{bmatrix}$$

- Matrix multiplication, and element-wise operations, are key tools in deep learning that enable efficient computation, manipulation, and transformations of data within neural networks.

Matrix operations

- **Matrix transpose**: Matrix transposition (often denoted by a superscript 'T' e.g. M^T) provides a way to "rotate" one of the matrices so that the operation complies with multiplication requirements and can continue.
- There are two steps to transpose a matrix:
  - Rotate the matrix right 90°
  - Reverse the order of elements in each row (e.g. [a b c] becomes [c b a])

As an example, transpose matrix M into T:

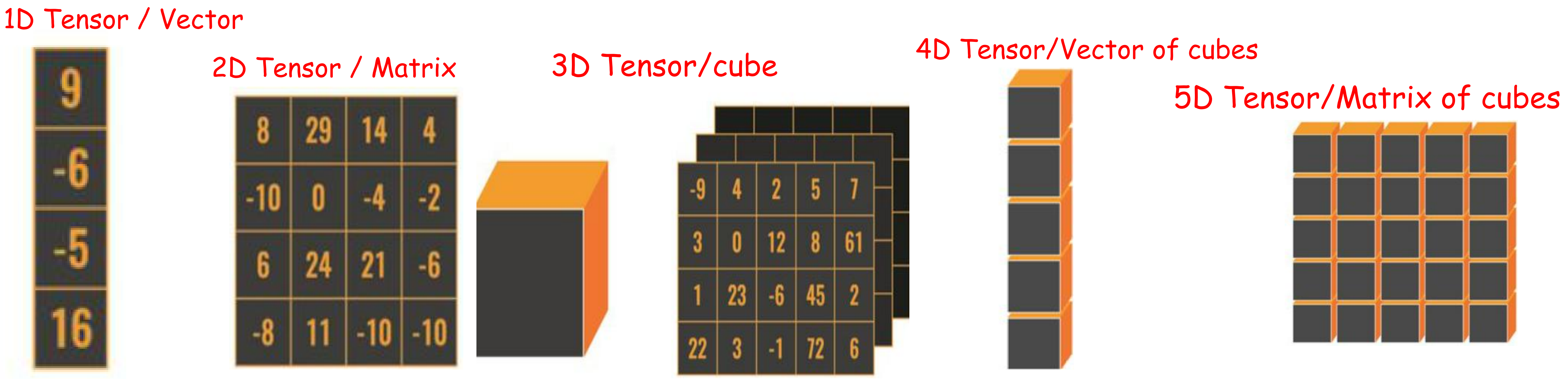
$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \Rightarrow \begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix}$$

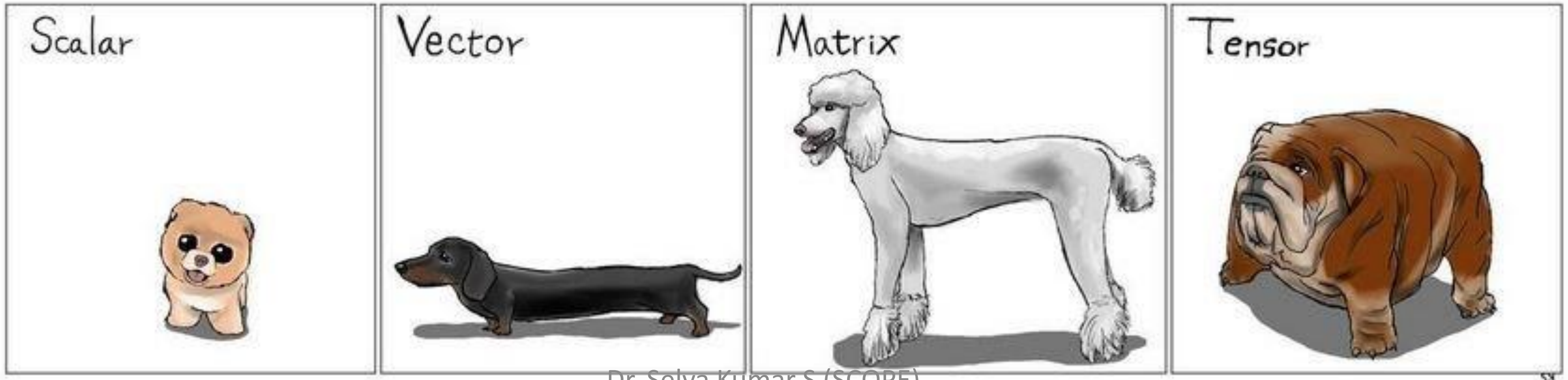
Tensors

- Tensors are multi-dimensional arrays that generalize the concept of vectors and matrices.
- In deep learning, tensors are a fundamental data structure used to represent and process data of various types and dimensions.
- Tensor : Extension of Matrix
- Here are some common tensor representations:
- Vectors: 1D — (features)
- Sequences: 2D — (timesteps, features)
- Images: 3D — (height, width, channels)
- Videos: 4D — (frames, height, width, channels)

Tensor Cont'd

- For example,





Tensor Cont’d

- Indexes required to access the element in tensor.

Indexes required	**Computer science**	**Mathematics**
0	number	scalar
1	array	vector
2	2d-array	matrix
n	nd-array	nd-tensor

Why Tensors?

Statistical reasons:

- Incorporate higher-order relationships in data.
- Discover hidden topics (not possible with matrix methods)

Computational reasons:

- Tensor algebra is parallelizable like linear algebra.
- Faster than other algorithms for Linear Discriminant Analysis (LDA)
- Flexible: Training and inference decoupled
- Guaranteed in theory to converge to the global optimum

Examples for tensors

- Images are commonly represented as tensors.
- A grayscale image can be represented as a 2D tensor, where each element represents the pixel intensity.
- Color images are represented as 3D tensors, with each element storing color information for each pixel (e.g., RGB channels).
- Text or time-series data, such as sentences or audio, can be represented as tensors.
- A sentence can be encoded as a 2D tensor, with each row representing a word and each column representing a feature.
- Time-series data can be represented as a 3D tensor, with dimensions representing time steps, features, and samples.
- The weights and biases in neural networks are represented as tensors. The dimensions of weight tensors depend on the architecture and layer sizes.
- These tensors store the learnable parameters that are updated during the training process.

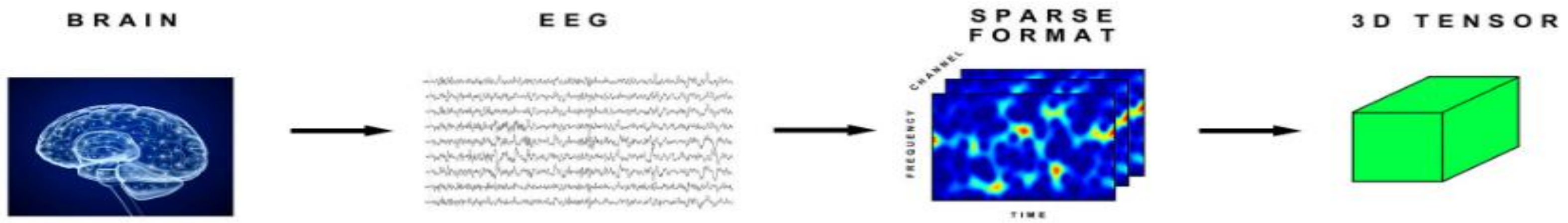
Example for vector data: 2D tensor data

- A statistical dataset of consumers, where each individual's age, height, and gender are taken into account. Since each individual may be represented as a vector of three values, the full dataset of 100 individuals can be stored in a 2D tensor of the shape (100, 3).

- A collection of textual information in which each article is represented by the number of times each word occurs in it (out of a dictionary of 2000 common words). A full dataset of 50 articles can be kept in a tensor of shape (50, 2000) since each article can be represented as a vector of 20,00 values (one count per word in the dictionary).

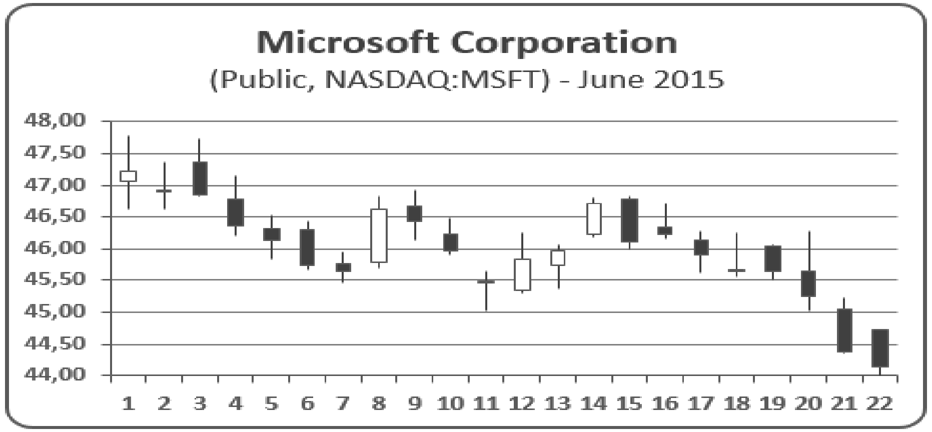
Example : Time series data or sequence data(3D tensor data

- Medical Scans - We can encode an electroencephalogram EEG signal from the brain as a 3D tensor, because it can be encapsulated as 3 parameters:



(time, frequency, channel)

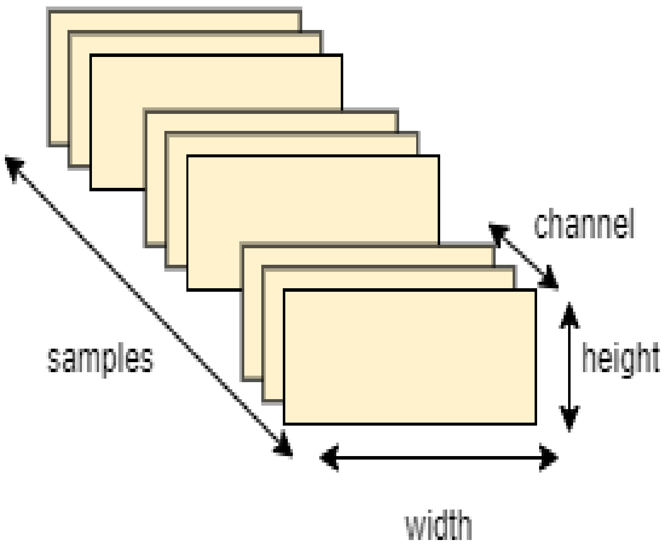
- Stock Prices -Stock prices have a high, a low and a final price every minute. The Mumbai Stock Exchange is open from 9:30 AM to 4 PM. That's 6 1/2 hours. There are 60 minutes in an hour so 6.5 x 60 = 390 minutes. These are typically represented by a candle stick graph.

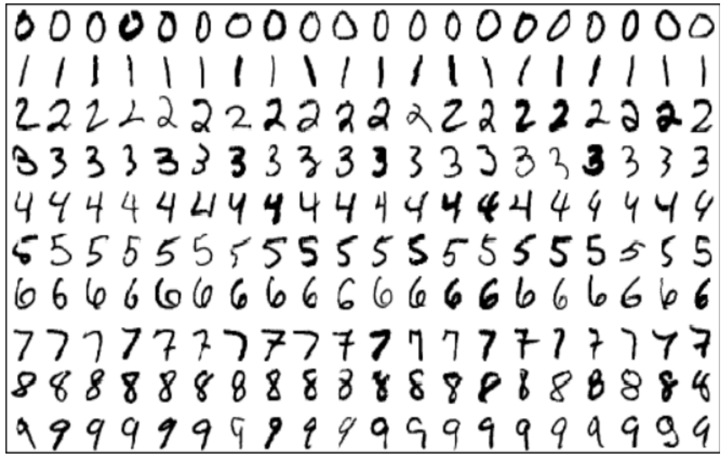


(week_of_data, minutes, high_low_price)

4D Tensor data

- 4D tensors are great at storing a series of images like Jpegs.
- The image is a 3D tensor, but the set of images makes it 4D. Remember that the fourth field is for sample_size.





- The famous MNIST data set is a series of handwritten numbers that stood as a challenge for many data scientists for decades, but are now considered a solved problem, with machines able to achieve 99% and higher accuracy.

Video data (5D Tensor)

- A video could be viewed as a set of coloured images called frames.
- A batch of various movies can be saved in a 5D tensor of shape (samples, frames, height, width, and colour-depth) since each frame can be kept in a 3D tensor (height, width, and colour-depth).
- A series of frames can also be saved in a 4D tensor (frames, height, width, and colour-depth).
- Example: 240 frames would be present in a 60-second, 144 x 256 YouTube video clip sampled at 4 frames per second. Four of these video clips would be saved in a tensor shape as a batch (4, 240, 144, 256, 3).

Tensor Operations

Addition

A normal matrix addition involves element-wise addition. Let us consider two matrices A and B and their sum resulting in a new matrix C.

$$A_{ij} + B_{ij} = C_{ij}$$

For addition of two matrices, the dimensions of the matrices must match. The final resulting matrix would also be of the same dimension.

Example:

If $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$, then the addition of the matrices would go like this:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix}$$

Tensor Operations

Multiplication

- Multiplication of a matrix *A* of dimension *m∗n* and a matrix *B* of dimension *n∗q* is given by:

$$C_{ij} = \sum_k A_{ik} B_{kj}$$

- For multiplication, the number of columns of the first matrix must match with the number of rows of the second matrix.

Example:

Let's suppose two matrices, $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ and $B = \begin{pmatrix} 1 & 2 \\ 3 & 1 \\ 2 & 3 \end{pmatrix}$

$$C_{ij} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} * \begin{pmatrix} 1 & 2 \\ 3 & 1 \\ 2 & 3 \end{pmatrix} = \begin{pmatrix} 1*1+2*3+3*2 & 1*2+2*1+3*3 \\ 4*1+5*3+6*2 & 4*2+5*1+6*3 \end{pmatrix}$$

$$C_{ij} = \begin{pmatrix} 13 & 13 \\ 31 & 31 \end{pmatrix}$$

- Multiplication of a matrix and a vector (column vector) is also possible until the dimension conditions match.

Tensor Operations

Hadamard Product

- The Hadamard product involves element-wise multiplication. Multiplying matrices must be of the same dimensions.

$$C_{ij} = A_{ij} \odot B_{ij}$$

Example:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \odot \begin{pmatrix} 2 & 2 & 3 \\ 4 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 4 & 9 \\ 16 & 10 & 6 \end{pmatrix}$$

Tensor Operations

Dot Product

- A dot product of two vectors results into a scalar. The dot product between vectors $\vec{x}$ and $\vec{y}$ is defined as (in vector notation).

$$\vec{x}\,.\,\vec{y} = a$$

$$\sum_i x_i y_i = a$$

where, a is scalar.

Example:

Let's suppose $\vec{x} = \begin{bmatrix}1 & 2 & 3 & 4\end{bmatrix}$ and $\vec{y} = \begin{bmatrix}2 & 3 & 1 & 4\end{bmatrix}$

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

$$= \begin{bmatrix}1*2+2*3+3*1+4*4\end{bmatrix} = 27$$

In matrix notation, the dot product is written as: $x^T * y = a$ Or $x * y^T = a$

where the superscript T denotes the transpose operation. The transpose makes sure that the two vectors are compatible for multiplication

Tensor Operations

Transpose

- The transpose of a matrix A which is denoted by $B=A^T$, is given by $B_{ji}=A_{ij}$.

Example:

Let's suppose a matrix $A = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \end{pmatrix}$. The transpose A^T is:

$$B = A^T = \begin{pmatrix} 1 & 3 \\ 2 & 4 \\ 3 & 5 \end{pmatrix}$$

Inverse

- A inverse matrix (A^{-1}) is a matrix that when multiplied with the original matrix (A) gives an identity matrix (I). An identity matrix is a special kind of square matrix that has ones in the diagonal with all other elements as zero.

$$I = A^{-1}A = AA^{-1}$$

An identity matrix I of dimension $3 * 3$ is defined as:

$$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Tensor Operations

Broadcasting

- This is a special type of tensor addition operation where a matrix and a vector are added. Earlier, we saw that the addition requires two tensors to have the same dimension for an element-wise sum.
- However, we already know that a vector is a 1-D array while a matrix is 2-D array. Therefore broadcasting is basically a programmatic approach of adding a matrix and a vector.
- The vector is automatically replicated to match the dimension of the matrix it is going to be added to.
- Here's the mathematical notation of broadcasting: $A_{ij}+vj=C_{ij}$, where A_{ij} is a matrix and v_j is a vector.
- The dimension of the vector, however, should match either the number of rows or the number of columns of the matrix it is going to be added to. Only this way, the vector can replicate itself for addition.

Example:

Let us suppose, $A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}, v = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

Although the dimensions do not match, the vector is replicated as a part of broadcasting and the following result is produced after addition:

$$C = \begin{pmatrix} 2 & 4 & 6 \\ 4 & 6 & 8 \end{pmatrix}$$

Tensor Concatenation

- Tensor concatenation is the process of combining tensors along a specified axis to form a larger tensor.
- This operation is commonly used to merge or combine tensors in different ways.
- Example:
- Tensor A: [[1, 2], [3, 4]]
- Tensor B: [[5, 6], [7, 8]]
- Resultant Tensor C (Concatenated along axis 0): [[1, 2], [3, 4], [5, 6], [7, 8]]

Differences between Matrices and Tensors:

	Matrices	Tensors
Dimensions	Two-dimensional	Arbitrary number of dimensions
Elements	Scalar elements	Elements can be scalars, vectors, matrices, or other tensors
Application	Used in linear algebra, systems of linear equations	Used in deep learning for representing and processing complex data structures
Rank	Rank 2	Rank can be higher than 2
Shape	Represented as (m, n)	Shape specified by the size along each dimension
Examples	[[1, 2], [3, 4]]	[[[1, 2], [3, 4]], [[5, 6], [7, 8]]]