Introduction

Vectors, Shapes, and Matrix Multiplication: Understand how vectors and matrices are defined and how to multiply them, focusing on the concept of dimensions and shape.

Concepts

Vector

A **vector** is a 1D array of numbers (e.g., a word representation). There are two types of vectors:

Row vector

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

Column vector

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

Does the orientation matter?

Mathematically: Yes. In Linear Algebra (and deep learning code like NumPy or PyTorch), a "Vector" is almost always assumed to be **vertical** (a column). If you try to multiply two vertical vectors together, the math will fail. You usually need to multiply a **Row** (Horizontal) by a **Column** (Vertical).

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \times \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} = 32$$

How the math works (Dot Product): Just for your reference, here is the breakdown of how that calculation happens to get **32**. You multiply the first element of the row by the first element of the column, add it to the second multiplied by the second, and so on:

$$(1 \times 4) + (2 \times 5) + (3 \times 6) =$$
$$4 + 10 + 18 = 32$$

The Dot Product is specifically a type of multiplication between two vectors that results in a single number **(a scalar)**.

However, in computer science (like Python/NumPy) and some linear algebra contexts, people often use the term loosely to refer to matrix multiplication as well.

The Strict Definition (Vectors) Strictly speaking, the dot product is an operation between two vectors of equal length.

- Input: Two vectors ($[a, b, c]$ and $[x, y, z]$).
- Operation: Multiply matching pairs and sum them up ($ax + by + cz$).
- Output: A single number (Scalar).
- Visual:

$$\mathbf{a} \cdot \mathbf{b} = \sum a_i b_i$$

The notation $a_i b_i$ is shorthand for **"the element at position i in vector a multiplied by the element at position i in vector b."**

It relies on the concept of an **Index** (represented by the letter i), which is just a fancy way of saying "position number."

Here is the step-by-step breakdown:

1. The Index (i)

Imagine our two vectors from before, $\mathbf{a}$ and $\mathbf{b}$.

- $\mathbf{a} = [1, 2, 3]$
- $\mathbf{b} = [4, 5, 6]$

The "Index" i counts the position:

- When $i = 1$: We are looking at the **1st** number in each list (1 and 4).
- When $i = 2$: We are looking at the **2nd** number in each list (2 and 5).
- When $i = 3$: We are looking at the **3rd** number in each list (3 and 6).

2. The Term $a_i b_i$

So, a_i just means "value of vector **a** at position **i**".

- If $i = 1$, then $a_1 = 1$ and $b_1 = 4$. Therefore, $a_1 b_1 = 1 \times 4 = 4$.
- If $i = 2$, then $a_2 = 2$ and $b_2 = 5$. Therefore, $a_2 b_2 = 2 \times 5 = 10$.

3. The Sigma Symbol ($\sum$)

The full formula usually looks like this:

$$\sum_{i=1}^{n} a_i b_i$$

The big Greek letter Sigma ($\sum$) means **"Sum"** (add them all up). The formula translates to English as:

> "Start at position 1 ($i = 1$). Multiply the pair at that position ($a_i b_i$). Keep doing this until you reach the last position (n). Then add all those results together."

Putting it together in your example

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

1. $i = 1$**:** Multiply first pair (1×4) = **4**
2. $i = 2$**:** Multiply second pair (2×5) = **10**
3. $i = 3$**:** Multiply third pair (3×6) = **18**

Sum ($\sum$): $4 + 10 + 18 = 32$

When you multiply matrices (or a row vector by a column vector, as we did before), it is often called the **Inner Product** or standard Matrix Multiplication.

- It is essentially calculating the **dot product** for every row of the first matrix against every column of the second matrix.

Matrix

A matrix is a 2D grid of numbers (e.g., a batch of sentences, or a weight matrix).

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Matrix Dimension

This is one of the most critical concepts when working with data in Python and Linear Algebra. If your dimensions don't match, your code will break.

Here is the breakdown of **Dimensions** (Math concept) vs. **Shape** (Code concept).

1. The Golden Rule: "Rows $\times$ Columns"

In both math and coding, we almost always list dimensions in this specific order: **Rows first, Columns second.**

A helpful mnemonic is **"RC"** (like RC Cola or Remote Control).

- **R**ows (Left-to-Right lines) come first.
- **C**olumns (Up-and-Down lines) come second.

2. Matrix Dimensions (Math Notation)

In mathematics, the "Dimension" describes the size of the matrix using the $m \times n$ notation.

- m: Number of rows.
- n: Number of columns.

If matrix A has **3 rows** and **2 columns**, we say:

> "A is a 3×2 matrix."

Visual:

$$\underbrace{\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}}_{3 \text{ rows}, 2 \text{ columns}}$$

3. Matrix Shape (Python/NumPy)

In Google Colab (using NumPy), we use the attribute `.shape` to see these dimensions. It returns a "Tuple" (a list of numbers in parentheses).

Math Dimension	NumPy Shape	Code Example
3×2 Matrix	`(3, 2)`	`[[1, 2], [3, 4], [5, 6]]`
2×3 Matrix	`(2, 3)`	`[[1, 2, 3], [4, 5, 6]]`
3×1 Vector	`(3, 1)`	`[[1], [2], [3]]` (Column)
1×3 Vector	`(1, 3)`	`[[1, 2, 3]]` (Row)

4. The "Rank 1" Trap (Important!)

There is a confusing aspect in Python called "Rank 1 Arrays."

If you create a simple list in NumPy, it doesn't have a row or column. It is just a flat list.

- **Shape:** `(3,)` ← Notice the missing number!
- **Behavior:** It can act weirdly because it is neither strictly vertical nor horizontal.

Recommendation: When doing serious Linear Algebra (like Deep Learning), always force your vectors to be **2D** (Rank 2) so you clearly have `(3, 1)` or `(1, 3)`.

5. Why Shape Matters (The "Inner Dimensions" Rule)

When multiplying two matrices, the **Inner Dimensions** must match.

If you multiply Matrix A $(3 \times \mathbf{2})$ by Matrix B $(\mathbf{2} \times 4)$:

1. Look at the numbers in the middle (the Inner Dimensions): **2** and **2**.
2. Do they match? **Yes.** (Multiplication works).
3. The result will take the **Outer Dimensions**: 3×4.

$$(3 \times \mathbf{2}) \cdot (\mathbf{2} \times 4) \rightarrow (3 \times 4)$$

If you try $(3 \times 2) \cdot (3 \times 2)$, the inner numbers are **2** and **3**. They don't match. **The code will error.**

What is d_{model}?

In pure mathematics, we talk about a vector having size n. In Deep Learning (specifically Transformers), we give that n a specific name: **`d_model`** (Dimension of the Model).

1. The Concept Imagine you want to represent the word "Guitar" to a computer. You can't just use the letters. You have to turn it into a list of numbers (a vector).

- If you turn "Guitar" into a list of **512 numbers**, then your `d_model` = 512.
- If you turn "Guitar" into a list of **4096 numbers** (like GPT-3), then your `d_model` = 4096.

2. Mapping it to Matrix Math Remember our matrix notation: **Rows** $\times$ **Columns**.

In a Transformer, data usually flows as a matrix where:

- **Rows** = The number of words in your sentence (Sequence Length).
- **Columns** = The richness of the description for each word (`d_model`).

Visual: If you have the sentence *"I play guitar"* (3 words) and your model uses a `d_model` of 4:

$$\text{Input Matrix} = \begin{bmatrix} 0.1 & -0.5 & 0.0 & 1.2 \\ 0.8 & 0.1 & -0.1 & 0.5 \\ -0.4 & 0.9 & 1.1 & -0.2 \end{bmatrix}$$

- **Shape:** (3×4)
- **Math Terms:** $3 \text{ rows} \times 4 \text{ columns}$
- **AI Terms:** $\text{Sequence Length} \times d_{model}$

3. Why does it matter? This is the "width" of the neural network.

- **Larger `d_model` :** The model can "think" about more complex concepts (it has 4096 numbers to describe "Guitar" instead of just 4).
- **Trade-off:** Matrix multiplications become much slower because the inner dimensions are massive.

Matrix multiplication

Matrix multiplication (also called the **Matrix Product**) is the process of combining two matrices to create a new one. It is the engine behind almost all modern neural networks.

It is **not** just multiplying the numbers in the same position (that is a different operation called "Element-wise" or "Hadamard" multiplication).

Here is how standard Matrix Multiplication works.

1. The Size Rule (Inner Dimensions)

As mentioned before, you can only multiply Matrix A by Matrix B if the **columns of A** equal the **rows of B**.

- A: $2 \times \mathbf{3}$
- B: $\mathbf{3} \times 2$
- Result: 2×2

If these "inner numbers" (3 and 3) don't match, the multiplication is mathematically impossible.

2. The Mechanism: "Row by Column"

To fill in a single spot in the new matrix, you perform a **Dot Product** between a **Row** from the first matrix and a **Column** from the second matrix.

The Algorithm: To find the number that goes in the i-th row and j-th column of the result:

1. Take the entire i**-th Row** from Matrix A.
2. Take the entire j**-th Column** from Matrix B.
3. Multiply their matching elements and add them up (Dot Product).

3. A Step-by-Step Example

Let's multiply two 2×2 matrices:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

Step 1: Top-Left (C_{11})

- Use **Row 1** of Left Matrix: $[1, 2]$
- Use **Column 1** of Right Matrix: $[5, 7]$
- Calculate: $(1 \times 5) + (2 \times 7) = 5 + 14 = \mathbf{19}$

Step 2: Top-Right (C_{12})

- Use **Row 1** of Left Matrix: $[1, 2]$
- Use **Column 2** of Right Matrix: $[6, 8]$
- Calculate: $(1 \times 6) + (2 \times 8) = 6 + 16 = \mathbf{22}$

Step 3: Bottom-Left (C_{21})

- Use **Row 2** of Left Matrix: $[3, 4]$
- Use **Column 1** of Right Matrix: $[5, 7]$
- Calculate: $(3 \times 5) + (4 \times 7) = 15 + 28 = \mathbf{43}$

Step 4: Bottom-Right (C_{22})

- Use **Row 2** of Left Matrix: $[3, 4]$
- Use **Column 2** of Right Matrix: $[6, 8]$
- Calculate: $(3 \times 6) + (4 \times 8) = 18 + 32 = \mathbf{50}$

Final Result:

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

4. Important Property: Order Matters

In standard multiplication (scalars), 2×3 is the same as 3×2. In Matrix multiplication, **this is not true.**

$$A \times B \neq B \times A$$

If you swap the order, you will get a completely different result (or an error if dimensions no longer align). This is why specific ordering in Deep Learning layers is so crucial.

Here is the rewritten chapter on Transposition, incorporating the definition, the notation, and the logic for deciding which matrix to flip.

Matrix Transposition

In Data Science and Linear Algebra, you will frequently encounter two matrices that contain the correct data but are "shaped" wrongly for multiplication. Transposition is the tool used to align them.

1. What is Transposition? Transposition is the act of flipping a matrix over its main diagonal.

- The **Rows** of the original matrix become the **Columns** of the new matrix.
- The **Columns** of the original matrix become the **Rows** of the new matrix.

Notation:

- **Math:** A^T (or sometimes A')
- **Python (NumPy):** `A.T`

Visual Example: If Matrix A has dimensions (3×2):

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

Then A^T has dimensions (2×3):

$$A^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

2. The Rules: Which one do I transpose?

When you have two matrices that don't fit together naturally, you don't pick a random one to transpose. You make the decision based on two specific checks.

The Scenario:

Imagine you have Matrix A (3×2) and Matrix B (3×2). You cannot multiply $A \times B$ because the inner dimensions (2 and 3) do not match.

Rule 1: The "Puzzle Piece" Check (Feasibility)

You must transpose one of them so that the **Inner Dimensions** match.

- If you transpose A, it becomes (2×3). Now it fits with B (3×2).
- If you transpose B, it becomes (2×3). Now A (3×2) fits with it.
- *Result:* In this specific case, both options satisfy the feasibility rule. So, we move to Rule 2.

Rule 2: The "Target Shape" Check (Outcome)

This is the deciding factor. Transposing different matrices results in different output shapes. You choose the one that gives you the dimensions you need for your specific problem.

Option A: Transpose the First ($A^T \times B$)

- **Math:** $(2 \times \mathbf{3}) \cdot (\mathbf{3} \times 2)$
- **Outcome:** A 2×2 Matrix.
- *When to use:* Use this if you want a smaller summary matrix comparing the features (columns) against each other.

Option B: Transpose the Second ($A \times B^T$)

- **Math:** $(3 \times \mathbf{2}) \cdot (\mathbf{2} \times 3)$
- **Outcome:** A 3×3 Matrix.
- *When to use:* Use this if you want a larger matrix comparing every sample (row) against every other sample.

3. Summary Cheat Sheet

Problem	Logic	Action
Dimensions don't match	Inner numbers must be equal.	Transpose one matrix so the touching numbers are identical.
Vectors ($x \cdot x$)	Cannot multiply col by col.	Transpose the first vector: $x^T x$ (Scalar result).
Result is too big/small	Outer numbers determine size.	Flip the matrix that puts the desired dimension on the "outside."

Homework

Here is a set of exercises designed to be solved on paper. This will help build your muscle memory for the "Row by Column" pattern.

Part 1: Theoretical Concepts *Answer these to test your understanding of dimensions and rules.*

1. **Dimension Check:** If Matrix A has shape $(3, 5)$ and Matrix B has shape $(5, 2)$, is the multiplication $A \times B$ possible? If so, what is the shape of the resulting matrix?

2. **The Trap:** If Matrix X is $(2, 3)$ and Matrix Y is $(2, 3)$, why can you **not** multiply $X \times Y$?

3. **Commutativity:** In standard algebra, $a \times b = b \times a$. Is this true for matrices? (i.e., does $A \times B = B \times A$?)

4. **Vocabulary:** What do we call a matrix filled with ones on the diagonal and zeros everywhere else (e.g., $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$)?

5. **Vector Output:** If you multiply a Matrix (4×3) by a Vector (3×1), is the result a Matrix, a Vector, or a Scalar?

Part 2: Calculation Exercises (10 Problems)

Write these down and solve them step-by-step.

1. The Basics (2×2)

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

2. Vector Multiplication (Dot Product style)

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

3. The "Impossible" Test

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

4. The Identity Matrix *Hint: This should require very little calculation if you recognize the pattern.*

$$\begin{bmatrix} 7 & 3 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

5. Rectangular Matrices (2×3 and 3×1)

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

6. Rectangular Matrices (2×3 and 3×2)

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

7. Negative Numbers

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

8. The "Permutation" (Swapping Rows)

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

9. Long Vectors (1×3 and 3×1)

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

10. Scalar Matrix (Scaling)

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \times \begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix}$$

Part 3: Transposition

Here are 3 exercises specifically designed to practice the logic of Transposition.

Exercises

1. The "Puzzle Piece" Logic (Feasibility) You have two matrices, A and B.

- Matrix A has dimensions (3×5).
- Matrix B has dimensions (3×5).

You cannot calculate $A \times B$ because the inner dimensions (5 and 3) do not match. **Question:** Write down the **two** valid equations using transposition that make multiplication possible. (State the dimensions of the result for each).

2. The "Target Shape" Logic (Outcome) You have two row vectors, P and Q.

- Vector P has dimensions (1×4).
- Vector Q has dimensions (1×4).

Question A: Which transposition equation ($P^T Q$ or $P Q^T$) results in a **Scalar** (single number)? **Question B:** Which transposition equation results in a 4×4 **Matrix**?

3. The "Neural Network" Scenario You have a dataset of inputs called X and a set of learned patterns called W.

- X dimensions: (100×10) [100 images, 10 pixels each]
- W dimensions: (5×10) [5 patterns, 10 pixels each]

You want to find out how well each of the **100 images** matches each of the **5 patterns**. This means you need a final output matrix of shape (100×5). **Question:** Write the multiplication equation (deciding which matrix to transpose) to get this (100×5) result.

Answer Key

Check your answers below only after you have tried solving them!

Part 1 Answers

1. **Yes.** The inner dimensions match (5). The result is $(3, 2)$.
2. **Dimension Mismatch.** Inner dimensions are 3 and 2. They do not match. You would need to transpose Y first.
3. **False.** Order matters significantly in linear algebra.
4. **The Identity Matrix** (usually denoted as I).
5. **A Vector** (specifically a 4×1 column vector).

Part 2 Answers

1. $\begin{bmatrix} 4 & 4 \\ 10 & 8 \end{bmatrix}$

2. **23** (Scalar)

3. **Impossible.** (2×2) vs (1×2). Inner dims (2 and 1) do not match.

4. $\begin{bmatrix} 7 & 3 \\ 2 & 1 \end{bmatrix}$ (The matrix stays the same).

5. $\begin{bmatrix} -2 \\ -2 \end{bmatrix}$

6. $\begin{bmatrix} 2 & 3 \\ 7 & 6 \end{bmatrix}$

7. $\begin{bmatrix} 7 & 4 \\ 3 & 0 \end{bmatrix}$

8. $\begin{bmatrix} 7 & 8 \\ 5 & 6 \end{bmatrix}$ (Notice it swapped the rows of the second matrix).

9. **3** (Scalar: $6 - 8 + 5$)

10. $\begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$ (It doubled every number).

Part 3 Answers

1. Answer:

- **Equation 1:** $A \times B^T$
  - Math: $(3 \times \mathbf{5}) \cdot (\mathbf{5} \times 3) = 3 \times 3$ Matrix.
- **Equation 2:** $A^T \times B$
  - Math: $(5 \times \mathbf{3}) \cdot (\mathbf{3} \times 5) = 5 \times 5$ Matrix.

2. Answer:

- **A (Scalar):** PQ^T
  - Math: $(1 \times \mathbf{4}) \cdot (\mathbf{4} \times 1) = (1 \times 1)$.
- **B (Matrix):** $P^T Q$

- Math: $(4 \times \mathbf{1}) \cdot (\mathbf{1} \times 4) = (4 \times 4)$.

3. Answer:

- **Equation:** $X \times W^T$
  - Math: $(100 \times \mathbf{10}) \cdot (\mathbf{10} \times 5) = (100 \times 5)$.
  - *Reasoning:* We kept X as is because we wanted the 100 to be the first dimension (rows) of the answer. We transposed W to make the inner 10s match.