Vector Algebra

Note by Shah Md. Arshad Rahman Ziban: This document provides an introduction to vector algebra and its application in solving linear equations efficiently using matrices. It includes examples and Python code snippets to illustrate the concepts.

Vector Algebra helps us solve equations fast and efficiently using matrices instead of long calculations.

Step 1: Given Data in Vector Form

We have the data set:

Chocolates (x)	**Energy (y)**
1	2
2	4
3	6
4	8

We can write it as matrices:

Feature Matrix X (with a bias column of 1)

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

Target Vector y

$$y = \begin{bmatrix} 2 \\ 4 \\ 6 \\ 8 \end{bmatrix}$$

```python
import numpy as np

# Step 1: Define X (with bias column) and y
X = np.array([
```

```
    [1, 1],
    [1, 2],
    [1, 3],
    [1, 4]
], dtype=float)

y = np.array([2, 4, 6, 8], dtype=float)  # Target values
```

Why is There a "1" in X?

The extra column of 1s is there to help us find the starting value of the line, called the intercept (c).

Think about a simple equation of a straight line:

$$y = mx + c$$

where:

- x is the input (chocolates),
- y is the output (energy),
- m tells us how much energy increases per chocolate,
- c is the starting value of energy, even if $x = 0$.

Example

Imagine a rule like:

$$\text{Energy} = 2 \times \text{Chocolates} + 1$$

Chocolates (x)	**Energy (y)**
1	3
2	5
3	7

Here, even if you eat 0 chocolates, your energy is still 1 because of c.

How Does the "1" Help in a Matrix?

If we write the equation as a matrix, we need to include c. We use a matrix like this:

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}$$

The 1s help multiply c, so we can write:

$$y = \begin{bmatrix} c + m(1) \\ c + m(2) \\ c + m(3) \\ c + m(4) \end{bmatrix}$$

This makes sure we calculate both m (slope) and c (intercept).

What If We Remove the "1"?

If we don't add the 1s, the equation becomes:

$$y = mx$$

- This forces the line to pass through (0,0).
- We lose the ability to find the intercept c.
- Our predictions might be wrong if there's an initial energy value!

Simple Analogy

Think of it like money:

- m is how much money you earn per day.
- c is the money you already have before starting work.

If you don't add the 1s, we forget the money we had at the start!

Final Answer

The extra "1" column in X helps us find both:

- m (increase per chocolate),
- c (starting energy when chocolates = 0).

Step 2: Compute β (m and c) Using Vector Algebra

The formula is:

$$\beta = (X^T X)^{-1} X^T y$$

Since $X^T X$ is a 2×2 matrix, we will compute its inverse manually. Then we will multiply it with $X^T y$ to get the best m and c.

```
# Compute X^T * X
XT_X = np.dot(X.T, X)

# Compute X^T * y
XT_y = np.dot(X.T, y)

# Compute the inverse of X^T * X manually (since it's a 2x2
    matrix)
a, b = XT_X[0, 0], XT_X[0, 1]
c, d = XT_X[1, 0], XT_X[1, 1]

det = a * d - b * c  # Determinant
adjugate = np.array([[d, -b], [-c, a]])  # Adjugate matrix

XT_X_inv = (1 / det) * adjugate  # Compute the inverse

# Compute beta = (X^T * X)^(-1) * (X^T * y)
beta = np.dot(XT_X_inv, XT_y)
beta
```

Mathematical Explanation

1. Compute $X^T X$ and $X^T y$:

$$X^T X = \begin{bmatrix} \sum_{i=1}^n 1 & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{bmatrix}$$

$$X^T y = \begin{bmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{bmatrix}$$

2. Compute the inverse of $X^T X$:

$$\text{Given } X^T X = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\text{Determinant: } \det(X^T X) = ad - bc$$

$$\text{Adjugate matrix: } \operatorname{adj}(X^T X) = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

$$(X^T X)^{-1} = \frac{1}{\det(X^T X)} \operatorname{adj}(X^T X) = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

3. Compute β:

$$\beta = (X^T X)^{-1} X^T y$$

OR

```
# Compute the best m and c
beta = np.linalg.inv(X.T @ X) @ X.T @ y
```

Prediction Section

The Role of $\beta = [0, 2]$

1. What Does β Represent?

The vector $\beta = [0, 2]$ represents the parameters of the linear equation.

- The first value $\beta_0 = 0$ corresponds to the intercept (c) in the equation.
- The second value $\beta_1 = 2$ corresponds to the slope (m) in the equation.

2. Equation Using β:

The linear equation we are using is:

$$y = \beta_0 + \beta_1 \cdot x$$

Substituting the values $\beta_0 = 0$ and $\beta_1 = 2$, the equation becomes:

$$y = 0 + 2 \cdot x$$

or simply:

$$y = 2x$$

This confirms that the slope is 2, and the intercept is 0.

Step 1: Prediction for New Values

We are predicting energy (y) for new values of chocolates (x):

$$x = [5, 10, 15]$$

We apply the equation $y = 2x$ to each value of x:

Step 2: Calculate Predictions

For each value of x, we calculate y:

1. For $x = 5$:
$$y = 2(5) = 10$$
2. For $x = 10$:
$$y = 2(10) = 20$$
3. For $x = 15$:
$$y = 2(15) = 30$$

Step 3: Code Explanation in Terms of β

```
def predict(x):
    return beta[1] * x + beta[0]  # beta[1] is the slope,
        beta[0] is the intercept
```

Explanation of the Predict Function

Where:

$$\beta[0] = 0 \quad \text{(intercept } c\text{)}$$

$$\beta[1] = 2 \quad \text{(slope } m\text{)}$$

So the predict function is calculating:

$$y = \beta[1] \cdot x + \beta[0]$$

Substituting $\beta = [0, 2]$:

$$y = 2x$$

Prediction Code

```
# Predict for new values
new_x_values = [5, 10, 15]  # Chocolates to predict for
predictions = [predict(x) for x in new_x_values]

# Display predictions
for x, y_pred in zip(new_x_values, predictions):
    print(f"For x = {x} chocolates, Predicted energy = {
        y_pred}")
```

Explanation

For $x = 5$:

$$y = 2 \cdot 5 = 10$$

For $x = 10$:

$$y = 2 \cdot 10 = 20$$

For $x = 15$:

$$y = 2 \cdot 15 = 30$$

So the predictions are:

$$y = [10, 20, 30]$$

Summary

The vector $\beta = [0, 2]$ represents the intercept and slope of the linear equation.

The prediction formula $y = 2x$ gives us the energy based on the number of chocolates.

We used this to predict energy for 5, 10, and 15 chocolates.