Matrix Multiplication - Code Along

Introduction

Understanding matrix operations is very important for a deeper understanding of linear algebra. We know matrices are used throughout the field of machine learning in the description of algorithms and representation of data. In this lesson, we shall discover how to manipulate matrices in Python and Numpy.

Objectives

You will be able to:

- Compute the dot product for matrices and vectors
- Calculate a cross product using Numpy
- Define a cross product

Definition

Multiplication of two matrices is one of the most crucial operations involving matrices. You can write the matrix product just by placing two or more matrices together, for example,

> $C = AB$

The standard product of two matrices is not just a matrix containing the element-wise product of the individual elements. This type of operation is a *special case* and is called the element-wise product, or the **Hadamard product**.

Hadamard product

Two matrices with the same dimensions can be multiplied together. Such element-wise matrix multiplication is called the Hadamard product. It's not the typical operation meant when referring to matrix multiplication, therefore a different operator is often used, such as a circle $\circ$.

> $C = A \circ B$

As with element-wise addition and subtraction, element-wise multiplication involves the multiplication of elements from each parent matrix to calculate the values in the new matrix as shown below.

$$A \circ B = \begin{bmatrix} A_{1,1} * B_{1,1} & A_{1,2} * B_{1,2} \\ A_{2,1} * B_{2,1} & A_{2,2} * B_{2,2} \\ A_{3,1} * B_{3,1} & A_{3,2} * B_{3,2} \end{bmatrix}$$

The Hadamard product can be calculated in Python using the $*$ operator between two NumPy arrays:

```
# Element-wise Hadamard product
import numpy as np
A = np.array([[1, 2, 3], [4, 5, 6]])
print(A)
B = np.array([[1, 2, 3], [4, 5, 6]])
print(B)
print ('\nHadamard product\n\n', A * B)
```

In [ ]:
```
# Code here
```

Dot product

The matrix dot product is more complicated than the previous operations and involves a rule as **not all matrices can be dot multiplied together**. The rule is as follows:

The matrix product of matrices A and B is a another matrix C. For defining this product, A must have the same number of dimensions as B has rows.

> When using the dot product, the number of columns in the first matrix must be equal the number of rows in the second matrix

For example, think of a matrix A having m rows and n columns and matrix B having n rows and k columns. Provided the n columns in A and n rows b are equal, the result is a new matrix with m rows and k columns. The dot product can be shown using (.) or (dot).

> $C_{(m,k)} = A_{(m,n)} \cdot B_{(n,k)}$ OR $C_{(m,k)} = A_{(m,n)} \text{ dot } B_{(n,k)}$

The product operation is defined by

$$C_{i,j} = \sum_k A_{i,k} B_{k,j}$$

The calculations are performed as follows:

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

$$B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$$

$$C = \begin{bmatrix} A_{1,1} * B_{1,1} + A_{1,2} * B_{2,1} & A_{1,1} * B_{1,2} + A_{1,2} * B_{2,2} \\ A_{2,1} * B_{1,1} + A_{2,2} * B_{2,1} & A_{2,1} * B_{1,2} + A_{2,2} * B_{2,2} \\ A_{3,1} * B_{1,1} + A_{3,2} * B_{2,1} & A_{3,1} * B_{1,2} + A_{3,2} * B_{2,2} \end{bmatrix}$$

This rule applies for a chain of matrix multiplications. The number of columns in one matrix in the chain must match the number of rows in the following matrix in the chain. The intuition for the matrix multiplication is that you calculate the dot product between each row in matrix A with each column in matrix B. For example, you can step down rows of column A and multiply each with column 1 in B to give the scalar values in column 1 of C.

This is made clear with the following worked example between two matrices.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} \times \begin{bmatrix} 2 & 7 \\ 1 & 2 \\ 3 & 6 \end{bmatrix} = \begin{bmatrix} 2*1+1*2+3*3 & 7*1+2*2+6*3 \\ 2*4+1*5+3*6 & 7*4+2*5+6*6 \\ 2*7+1*8+3*9 & 7*7+2*8+6*9 \\ 2*10+1*11+3*12 & 7*10+2*11+6*12 \end{bmatrix} = \begin{bmatrix} 13 & 29 \\ 31 & 74 \\ 49 & 119 \\ 67 & 164 \end{bmatrix}$$

Let's define above matrices and see how to achieve this in Python and Numpy using the `.dot()` method :

```
# matrix dot product
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])
B = np.array([[2, 7], [1, 2], [3, 6]])

C = A.dot(B)

print(A, '\ndot', '\n', B, '\n = \n', C)
```

In [ ]:
```
# Code here
```

Matrix-vector dot product

A matrix and a vector can be multiplied together as long as the rule of matrix multiplication (stated above) is observed. The number of columns in the matrix must equal the number of rows in the vector. As with matrix multiplication, the operation can be written using the dot notation. Because the vector only has one column, the result is always a vector. See the general approach below where A is the matrix being multiplied to v, a vector:

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

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$$C = \begin{bmatrix} A_{1,1} * v_1 + A_{1,2} * v_2 \\ A_{2,1} * v_1 + A_{2,2} * v_2 \\ A_{3,1} * v_1 + A_{3,2} * v_2 \end{bmatrix}$$

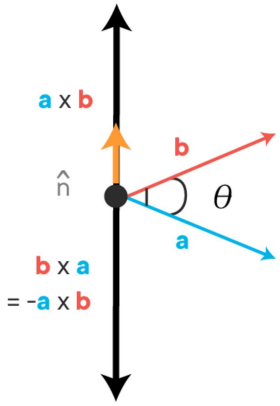
The matrix-vector multiplication can be implemented in NumPy using the `.dot()` method as seen before:

```
# matrix-vector multiplication

A = np.array([[1, 2], [3, 4], [5, 6]])
v = np.array([0.5, 0.5])

C = A.dot(v)

print(A, 'dot', v, f'= {C}', sep='\n')
```

In [ ]:
```
# Code here
```

Cross product

From basic geometry, you know that a vector has magnitude (how long it is) and direction. Two vectors can be multiplied using the "cross product". The cross product or vector product is a binary operation on two vectors in three-dimensional space. The result is a vector which is perpendicular to the vectors being multiplied and normal to the plane containing them.



The cross product of two vectors a and b is denoted by $(a \times b)$.

It's defined as:

$$a \times b = \| a \| \| b \| \sin(\theta) n$$

- $|a|$ is the magnitude (length) of vector a
- $|b|$ is the magnitude (length) of vector b
- θ is the angle between a and b
- n is the unit vector at right angles to both a and b

If either of the vectors being multiplied is zero or the vectors are parallel then their cross product is zero. More generally, the magnitude of the product equals the area of a parallelogram with the vectors as sides. If the vectors are perpendicular the parallelogram is a rectangle and the magnitude of the product is the product of their lengths.

In Numpy, you can take a cross product with `np.cross()` function:

```
# Cross product between two vectors
x = np.array([0, 0, 1])
y = np.array([0, 1, 0])

print(np.cross(x, y))
print(np.cross(y, x))
```

In [ ]:
```
# Code here
```

You'll look at the applications of the cross product later, when studying machine learning algorithms and developing geometric intuitions. In Natural Language Processing, cross products are also important to bring text into vector space, and check for document similarity. For now we'll look a bit more into the dot product.

Summary

In this lesson, you learned about matrix multiplication using the Hadamard product and the dot product. You also looked at how to take the dot product between vectors and matrices while observing the size assumptions. You also saw how cross products work between two vectors. Next, you'll learn about some properties of dot products and what makes them so special.