Linear_Algebra

December 16, 2022

1 Linear Alzebra

- Linear Alzebra
- Dot Product
- Determinant
- Inverse
- Eigenvalues and eigenvectors
- Norm
- Cos Theta, Tan Theta

[1]:
```
import numpy as np
np.__version__
```

[1]:
```
'1.21.5'
```

2 Linear Alzebra

[2]:
```
arr1 = np.array(
    [
        [10, 15, 20],   # 1st Index
        [70, 76, 89],   # 2nd Index
        [52, 45, 78]    # 3rd Index
    ]
)
```

3 Import Linear Alzebra from Numpy

- Here linage/ linear alzebra is a sub module imported from numpy

[3]:
```
from numpy import linalg as lg
```

[4]:
```
arr1
```

[4]:
```
array([[10, 15, 20],
       [70, 76, 89],
       [52, 45, 78]])
```

- In mathematics, the determinant is a scalar value that is a function of the entries of a square matrix. It characterizes some properties of the matrix and the linear map represented by the matrix.

[5]:
```
lg.det(arr1)   # Determinant of arr
```

[5]:
```
-9290.000000000015
```

[6]:
```
lg.inv(arr1)  # inverse
```

[6]:
```
array([[-0.20699677,  0.02906351,  0.01991389],
       [ 0.08955867,  0.02798708, -0.05489774],
       [ 0.08632939, -0.03552207,  0.03121636]])
```

[7]:
```
arr1.transpose()  # Transpose, e.g it will convert row to column & column to row
```

[7]:
```
array([[10, 70, 52],
       [15, 76, 45],
       [20, 89, 78]])
```

[8]:
```
arr1.T  # alternative of transpose
```

[8]:
```
array([[10, 70, 52],
       [15, 76, 45],
       [20, 89, 78]])
```

4 Dot Product Operation

- Need two matrix to operate dot product

[9]:
```
arr1
```

[9]:
```
array([[10, 15, 20],
       [70, 76, 89],
       [52, 45, 78]])
```

[10]:
```
arr2 = np.array(
    [
        [20, 30, 40],
        [15, 20, 25],
        [30, 35, 40]
    ]
)
```

4.0.1 3 x 3 matrix dot product

Basic Idea

- 1st row (matrix 1) x 1st columns (matrix 2)

- 1st row (matrix 1) x 2nd columns (matrix 2)
- 1st row (matrix 1) x 3rd columns (matrix 2)

[11]:
```
np.dot(arr1,arr2)  # Dot Product
```

[11]:
```
array([[1025, 1300, 1575],
       [5210, 6735, 8260],
       [4055, 5190, 6325]])
```

Dot Product Verification

[12]:
```
(10*20) + (15*15) + (20*30)    # 1st row of array1 * 1st columns of array 2  =
↪1025 verified
```

[12]:
```
1025
```

[13]:
```
(10*30) + (15*20) + (20*35)    # 1st row of array1 * 2 nd columns of array 2  =
↪1300 verified
```

[13]:
```
1300
```

[14]:
```
(10*40) + (15*25) + (20*40)    # 1st row of array1 * 3 rd columns of array 2  =
↪1575 verified
```

[14]:
```
1575
```

5 Dot Product of Two Vector

- Given Two Vectory, vector A & vector B, Find dot product of vector A & vector B

[15]:
```
a = [4, 1, 2, 3]  # Vector A
```

[16]:
```
b = [3, 1, 7, 2] # Vector B
```

[17]:
```
np.dot(a,b)
```

[17]:
```
33
```

Dot Product Calucation of Two Vector (A & B) shown as details/ working

[18]:
```
(4 * 3) + (1 * 1) + (2 * 7) + (3 * 2)   # Dot Product Calculation of two vector
↪is verifed
```

[18]:
```
33
```

6 Eigenvalues and eigenvectors

- The eigenvector is a vector that is associated with a set of linear equations. The eigenvector of a matrix is also known as a latent vector, proper vector, or characteristic vector

- In linear algebra, an eigenvector or characteristic vector of a linear transformation is a nonzero vector that changes at most by a scalar factor when that linear transformation is applied to it.
- The corresponding eigenvalue, often denoted by , is the factor by which the eigenvector is scaled.

[19]:
```
lg.eig(arr1)  # Eigenvector of normal vertor .eg arr1
```

[19]:
```
(array([154.73900897,  -4.39602447,  13.6570155 ]),
 array([[ 0.16279192,  0.86253644, -0.04101419],
        [ 0.79901468, -0.37530894, -0.80291219],
        [ 0.57885606, -0.33937308,  0.59468467]]))
```

6.0.1 Norm

- linalg.norm(x, ord=None, axis=None, keepdims=False)
- numpy.linalg.norm
- Matrix or vector norm
- This function is able to return one of eight different matrix norms, or one of an infinite number of vector norms (described below), depending on the value of the ord parameter.

[21]:
```
lg.norm(arr2) # its important used in algorithm
```

[21]:
```
88.74119674649424
```

[22]:
```
np.cos(90)   # cos theta of a singular value
```

[22]:
```
-0.4480736161291701
```

[23]:
```
np.cos(arr1) # Cos theta of an array/ cos theta of each value / list
```

[23]:
```
array([[-0.83907153, -0.75968791,  0.40808206],
       [ 0.6333192 ,  0.82433133,  0.51017704],
       [-0.16299078,  0.52532199, -0.85780309]])
```

[24]:
```
np.tan(90)  # tan theta of a singular value
```

[24]:
```
-1.995200412208242
```

[25]:
```
np.tan(arr1)
```

[25]:
```
array([[ 0.64836083, -0.8559934 ,  2.23716094],
       [ 1.22195992,  0.68674769,  1.68582537],
       [-6.05327238,  1.61977519, -0.59918   ]])
```