Matrices_plotting

August 17, 2020

0.1 Matrices

Transposing $\begin{pmatrix} a & b \\ c & d \end{pmatrix}^T = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$

$\begin{pmatrix} 1 & 5 & 7 \\ 2 & 4 & 8 \end{pmatrix}^T = \begin{pmatrix} 1 & 2 \\ 5 & 4 \\ 7 & 8 \end{pmatrix}$

$\begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}^T = \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}$

$\begin{pmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \end{pmatrix}^T = ?$

Matrix multiplication $\begin{pmatrix} 1 & 5 & 7 \\ 2 & 4 & 8 \end{pmatrix}$ x $\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1*1+5*0+7*1 & 1*0+5*1+7*1 \\ 2*1+4*0+8*1 & 2*0+4*1+8*1 \end{pmatrix} = \begin{pmatrix} 8 & 12 \\ 10 & 12 \end{pmatrix}$

Hint: Think about the steps you take when solving it. This will help when you need to code it.

$\begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$ x $\begin{pmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \end{pmatrix} = ?$

$\begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$ x $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 6 & 7 \end{pmatrix} = ?$

Inverting $\det\begin{pmatrix} a & b \\ c & d \end{pmatrix} = a*c - b*d$

$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \begin{pmatrix} d/det & -b/det \\ -c/det & a/det \end{pmatrix}$

$\det\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = ?$

$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}^{-1} = ?$

0.2 Misc

List comprehension

[1]:
```
my_list = [i for i in range(10)]
my_list
```

[1]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[2]:
```
my_second_list = [i**2 for i in range(5)]
my_second_list
```

[2]:
```
[0, 1, 4, 9, 16]
```

[3]:
```
# ToDo: creating a matrix of zeros
```

Modulus

[4]:
```
4 % 2
```

[4]:
```
0
```

[5]:
```
1 % 2
```

[5]:
```
1
```

[6]:
```
18 % 16
```

[6]:
```
2
```

Debugging

[7]:
```
# ToDo: pycharm demo
```

0.3 Numpy

[8]:
```
import numpy as np
```

[9]:
```
m1 = [[1,5,7], [2,4,8]]
np_m1 = np.array(m1)
np_m1
```

[9]:
```
array([[1, 5, 7],
       [2, 4, 8]])
```

[10]:
```
np_m1_t = np.transpose(np_m1)
np_m1_t
```

[10]:
```
array([[1, 2],
       [5, 4],
       [7, 8]])
```

[11]:
```
m2 = [[1,5], [2,4]]
np_m2 = np.array(m2)
np_m2
```

[11]:
```
array([[1, 5],
       [2, 4]])
```

[12]:
```
np.matmul(np_m1, np_m2)
```

```
␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-12-f1fa8e4719d2> in <module>
   ----> 1 np.matmul(np_m1, np_m2)


       ValueError: matmul: Input operand 1 has a mismatch in its core dimension␣
↪0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 2 is different from 3)
```

What is the error above? What kind of matrices can be multiplied with each other?

[13]:
```
np.shape(np_m1)
```

[13]:
```
(2, 3)
```

[14]:
```
np.shape(np_m1_t)
```

[14]:
```
(3, 2)
```

[15]:
```
np.shape(np_m2)
```

[15]:
```
(2, 2)
```

[16]:
```
# (n, k) * (k, m) = (n, m)
np.matmul(np_m1_t, np_m2)
```

[16]:
```
array([[ 5, 13],
       [13, 41],
       [23, 67]])
```

[17]:
```
np.linalg.inv(np_m2)
```

[17]:
```
array([[-0.66666667,  0.83333333],
       [ 0.33333333, -0.16666667]])
```

[18]:
```
np.linalg.inv(np_m1_t)
```

```
␣
↪---------------------------------------------------------------------------

       LinAlgError                               Traceback (most recent call␣
↪last)
```

```
    <ipython-input-18-fe804616ba4e> in <module>
----> 1 np.linalg.inv(np_m1_t)


    <__array_function__ internals> in inv(*args, **kwargs)


    ~/anaconda3/lib/python3.7/site-packages/numpy/linalg/linalg.py in inv(a)
    540     a, wrap = _makearray(a)
    541     _assert_stacked_2d(a)
--> 542     _assert_stacked_square(a)
    543     t, result_t = _commonType(a)
    544


    ~/anaconda3/lib/python3.7/site-packages/numpy/linalg/linalg.py in␣
↪_assert_stacked_square(*arrays)
    211         m, n = a.shape[-2:]
    212         if m != n:
--> 213             raise LinAlgError('Last 2 dimensions of the array must␣
↪be square')
    214
    215 def _assert_finite(*arrays):


    LinAlgError: Last 2 dimensions of the array must be square
```

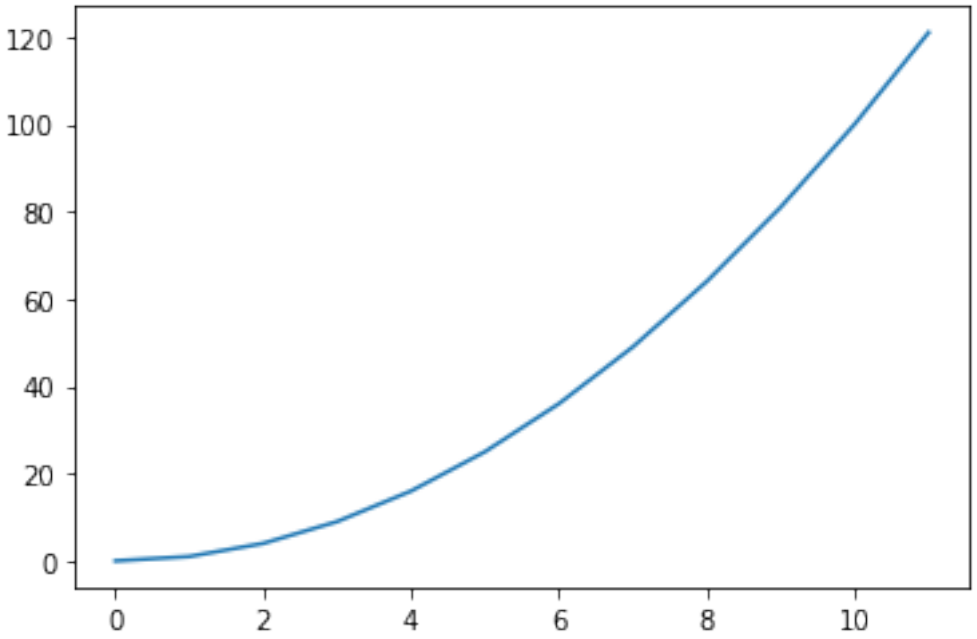
What is the error above? What type of matrix is invertible?

0.4 Matplotlib

[19]:
```
import matplotlib.pyplot as plt
```

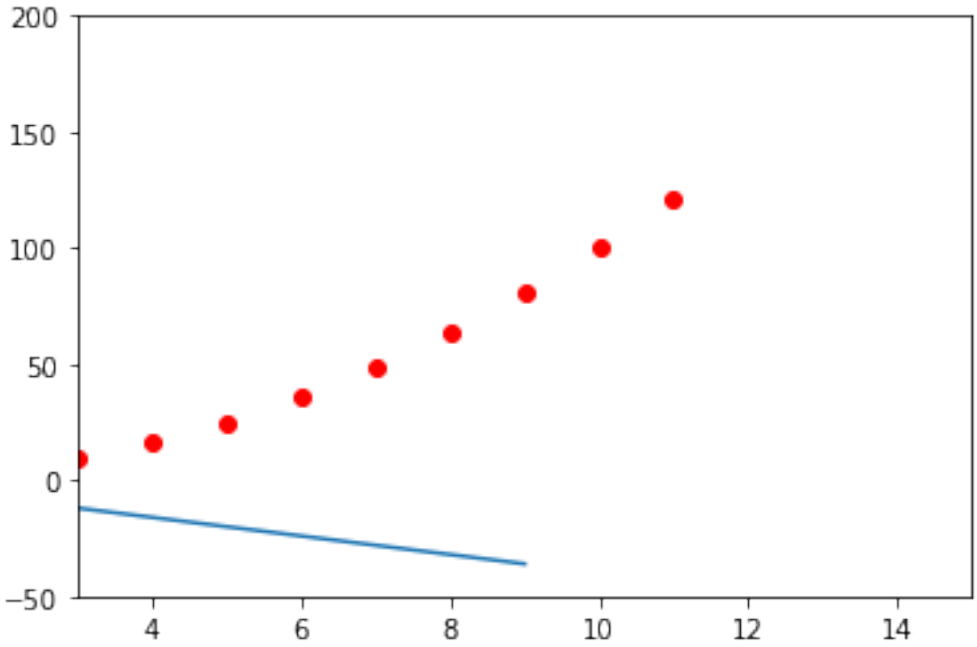
[20]:
```
a = [i for i in range(12)]
b = [i**2 for i in range(12)]
plt.plot(a, b)
```

[20]:
```
[<matplotlib.lines.Line2D at 0x7fb0944e6f60>]
```



[21]:
```
plt.plot(a, b, "ro")
plt.plot(my_list, [-i*4 for i in my_list])
plt.ylim(-50, 200)
plt.xlim(3, 15)
```

[21]: (3, 15)



0.5 Extra

BigO

[22]:
```
n = 5
for i in range(n):
    for j in range(n):
        print(i*j)
```

```
0
0
0
0
0
0
1
2
3
4
0
2
4
6
8
0
3
6
9
12
0
4
8
12
16
```

time complexity is N^2

[23]:
```
sum = 0
for i in range(n):
    for j in range(n):
        for k in range(n):
            sum += i*j*k
print(sum)
```

```
1000
```

time complexity is N^3

[24]:
```
# Code to print the list
def printList(arr):
        for i in range(len(arr)):
                print(arr[i], end =" ")
        print()
```

time complexity?

[25]:
```
# Python program for implementation of Bubble Sort

def bubbleSort(arr):
        n = len(arr)

        # Traverse through all array elements
        for i in range(n-1):
        # range(n) also work but outer loop will repeat one time more than␣
 ↪needed.

                # Last i elements are already in place
                for j in range(0, n-i-1):

                        # traverse the array from 0 to n-i-1
                        # Swap if the element found is greater
                        # than the next element
                        if arr[j] > arr[j+1] :
                                arr[j], arr[j+1] = arr[j+1], arr[j]
```

time complexity?

[26]:
```
# Python program for implementation of Insertion Sort

# Function to do insertion sort
def insertionSort(arr):

        # Traverse through 1 to len(arr)
        for i in range(1, len(arr)):

                key = arr[i]

                # Move elements of arr[0..i-1], that are
                # greater than key, to one position ahead
                # of their current position
                j = i-1
                while j >=0 and key < arr[j] :
                                arr[j+1] = arr[j]
                                j -= 1
                arr[j+1] = key

# This code is contributed by Mohit Kumra
```

time complexity?

[27]:
```
# Python program for implementation of MergeSort

def mergeSort(arr):
        if len(arr) >1:
                mid = len(arr)//2 # Finding the mid of the array
                L = arr[:mid] # Dividing the array elements
                R = arr[mid:] # into 2 halves

                mergeSort(L) # Sorting the first half
                mergeSort(R) # Sorting the second half

                i = j = k = 0

                # Copy data to temp arrays L[] and R[]
                while i < len(L) and j < len(R):
                        if L[i] < R[j]:
                                arr[k] = L[i]
                                i+= 1
                        else:
                                arr[k] = R[j]
                                j+= 1
                        k+= 1

                # Checking if any element was left
                while i < len(L):
                        arr[k] = L[i]
                        i+= 1
                        k+= 1

                while j < len(R):
                        arr[k] = R[j]
                        j+= 1
                        k+= 1

# This code is contributed by Mayank Khanna
```

time complexity? (this one is tricky)

[28]:
```
# driver code to test the above code
if __name__ == '__main__':
    arr = [12, 11, 13, 5, 6, 7]
    print ("Given array is", end ="\n")
    printList(arr)

    bubbleSort(arr)
    print("Sorted by bubblesort: ", end ="\n")
    printList(arr)
```

```
    arr = [12, 11, 13, 5, 6, 7]
    insertionSort(arr)
    print("Sorted by insertionsort: ", end ="\n")
    printList(arr)

    arr = [12, 11, 13, 5, 6, 7]
    mergeSort(arr)
    print("Sorted by mergesort: ", end ="\n")
    printList(arr)
```

```
Given array is
12 11 13 5 6 7
Sorted by bubblesort:
5 6 7 11 12 13
Sorted by insertionsort:
5 6 7 11 12 13
Sorted by mergesort:
5 6 7 11 12 13
```

time complexity?