# hw1

October 8, 2022

[16]:
```python
import numpy as np
```

## 0.1 1. Matrix Multiplication

### 0.1.1 a)

$$\begin{bmatrix} 2500 & 350 & 200 \\ 2000 & 405 & 250 \\ 2000 & 325 & 400 \\ 2000 & 210 & 450 \end{bmatrix}$$

In this matrix the rows represent each month (September, October, November, and December respectively) and the columns represent each category of spending (housing, food, recreation/transportation).

### 0.1.2 b)

$$\begin{bmatrix} 2500 & 350 & 200 \\ 2000 & 405 & 250 \\ 2000 & 325 & 400 \\ 2000 & 210 & 450 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} =$$

$$\begin{bmatrix} 2500 * 1 + 350 * 1 + 200 * 1 \\ 2000 * 1 + 405 * 1 + 250 * 1 \\ 2000 * 1 + 325 * 1 + 400 * 1 \\ 2000 * 1 + 210 * 1 + 450 * 1 \end{bmatrix} =$$

$$\begin{bmatrix} 3050 \\ 2655 \\ 2725 \\ 2660 \end{bmatrix}$$

So costs for September, October, November and December were 3050, 2655, 2725, and 2660 respectively.

### 0.1.3 c)

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2500 & 350 & 200 \\ 2000 & 405 & 250 \\ 2000 & 325 & 400 \\ 2000 & 210 & 450 \end{bmatrix} =$$

$$\begin{bmatrix} 2500*1 & 350*1 & 200*1 \\ + & + & + \\ 2000*1 & 405*1 & 250*1 \\ + & + & + \\ 2000*1 & 325*1 & 400*1 \\ + & + & + \\ 2000*1 & 210*1 & 450*1 \end{bmatrix} =$$

$$\begin{bmatrix} 8500 & 1290 & 1300 \end{bmatrix}$$

So costs for housing, food, and recreation/transportation are 8500, 1290, and 1300 respectively.

### 0.1.4 d)

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2500 & 350 & 200 \\ 2000 & 405 & 250 \\ 2000 & 325 & 400 \\ 2000 & 210 & 450 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2500*1+350*1+200*1 \\ 2000*1+405*1+250*1 \\ 2000*1+325*1+400*1 \\ 2000*1+210*1+450*1 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 3050 \\ 2655 \\ 2725 \\ 2660 \end{bmatrix} =$$

$$\begin{bmatrix} 3050*1+2655*1+2725*1+2660*1 \end{bmatrix} = 11090$$

Total costs are 11090

### 0.1.5 e)

[17]:
```
# Our expenses matrix
expenses_matrix = np.array([
        [2500, 350, 200],
        [2000, 405, 250],
        [2000, 325, 400],
        [2000, 210, 450]
    ])

# A 3 by 1 vector
by_month = np.array([[1], [1], [1]])

# costs per month
expenses_matrix@by_month
```

```
[17]: array([[3050],
             [2655],
             [2725],
             [2660]])
```

```python
[18]: # costs per category

      # A 1 by 4 matrix
      by_cat = np.array([1,1,1,1])

      by_cat@expenses_matrix
```

```
[18]: array([8500, 1290, 1300])
```

```python
[19]: # All costs
      by_cat@expenses_matrix@by_month
```

```
[19]: array([11090])
```

## 0.2 2.

### 0.2.1 a)

The vector (0, 0, 0, 1, 0) would get the fourth row:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 8 & 0 & 1 & 1 \\ 9 & 2 & 9 & 4 \\ 1 & 5 & 9 & 9 \\ 9 & 9 & 4 & 7 \\ 6 & 9 & 8 & 9 \end{bmatrix} =$$

$$\begin{bmatrix} 8*0 & 0*0 & 1*0 & 1*0 \\ + & + & + & + \\ 9*0 & 2*0 & 9*0 & 4*0 \\ + & + & + & + \\ 1*0 & 5*0 & 9*0 & 9*0 \\ + & + & + & + \\ 9*1 & 9*1 & 4*1 & 7*1 \\ + & + & + & + \\ 6*0 & 9*0 & 8*0 & 9*0 \end{bmatrix} =$$

$$\begin{bmatrix} 9 & 9 & 4 & 7 \end{bmatrix}$$

### 0.2.2 b)

A vector of length 5 with all 0s except for its kth entry which should be a 1 instead

### 0.2.3 c)

A vector of length five with all 0s except for the scalar $a$ in the kth row and the scalar $b$ in the jth row

### 0.2.4 d)

The vector (0, 0, 1, 0) would get the third column

### 0.2.5 e)

A vector of length 4 with all 0s except for its kth entry which should be a 1 instead

### 0.2.6 f)

A vector of length four with all 0s except for the scalar $a$ in the kth row and the scalar $b$ in the jth row

### 0.2.7 (g)

[20]:
```
matrix2 = np.array([
    [8, 0, 1, 1],
    [9, 2, 9, 4],
    [1, 5, 9, 9],
    [9, 9, 4, 7],
    [6, 9, 8, 9]
])
```

[21]:
```
# To get the fourth row
np.array([0, 0, 0, 1, 0])@matrix2
```

[21]:
```
array([9, 9, 4, 7])
```

[22]:
```
# Get kth row
def get_kth_row(k, matrix):
    m,n = matrix.shape
    v = np.zeros(m)
    v[k - 1] = 1
    return v@matrix

# get the 3rd row
get_kth_row(3, matrix2)
```

[22]:
```
array([1., 5., 9., 9.])
```

[23]:
```
# Get kth row times a plus jth row times b
def row_combination(a,b,k,j,matrix):
    m,n = matrix.shape
    v = np.zeros(m)
```

```
    v[k - 1] = a
    v[j - 1] = b
    return v@matrix

# get 2 times first row plus 3 times second row
row_combination(2,3,1,2, matrix2)
```

[23]: array([43.,  6., 29., 14.])

[24]:
```
# To get the fourth column
matrix2@np.array([0,0,1,0])
```

[24]: array([1, 9, 9, 4, 8])

[25]:
```
# Get kth column
def get_kth_col(k, matrix):
    m,n = matrix.shape
    v = np.zeros(n)
    v[k - 1] = 1
    return matrix@v

# get the 2nd column
get_kth_col(2, matrix2)
```

[25]: array([0., 2., 5., 9., 9.])

[26]:
```
# Get kth columna times a plus jth column times b
def col_combination(a,b,k,j,matrix):
    m,n = matrix.shape
    v = np.zeros(n)
    v[k - 1] = a
    v[j - 1] = b
    return matrix@v

# get 5 times third col plus 2 times fourth col
col_combination(5,2,3,4, matrix2)
```

[26]: array([ 7., 53., 63., 34., 58.])

### 0.3 3.

The matrix is rank 1 because all the columns (and rows) are linear combinations of each other. If the columns are the vectors $x_1, x_2, x_3$ then $x_2 = 2x_1$ and $x_3 = 3x_1$

### 0.4 4.

The line would be where $w^T x_0 = 0$ or in this case $w_1 x_1 + w_2 x_2 + w_3 = 0$ or $ 3x_1 + 5x_2 - 2 = 0$ which when put into slope intercept form gives us $x_2 = -3/5x_1 + 2/5$. Everything above this line would have a +1 label on or below it has a -1 label. For instance the point 1, 2 would give us an 11 which is above 0 and there for a +1 label.

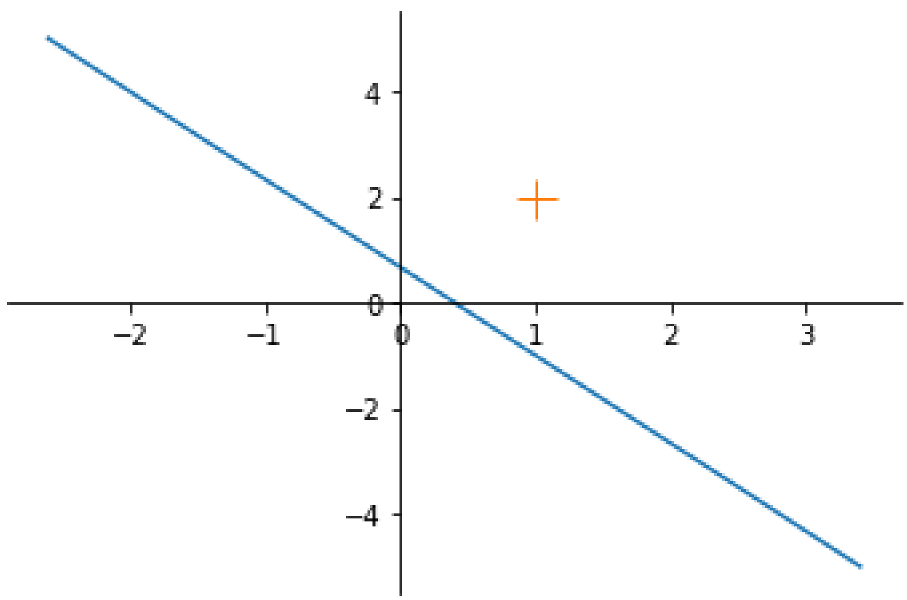
[27]:
```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.spines['top'].set_color('none')

x = np.linspace(-5, 5)

ax.plot((-3 * x + 2)/5, x, linestyle='-')
ax.plot(1, 2, marker="+", markersize=15)
```

[27]:
```
[<matplotlib.lines.Line2D at 0x1150e5dc0>]
```




## 0.5 5.

### 0.5.1 a)

You'd want to have d + 1 coefficients (an extra one that would be a constant for z to the 0th power or in otherwords 1). So you would have w_j from j=0 to j=d and z to the dth power for each coefficient ie:

$p(z) = y$

$\sum_{j=0}^{d} w_j z^d = y$

So if d=3:

$w_0 z^0 + w_1 z^1 + w_2 z^2 + w_3 z^3 = y$

### 0.5.2 b)

X should be a n by d+1 vector with each row being z_i to the power of 0...d then multipled by a vector of weights length d + 1 producing a y vector of length n (one for z) so:

$$\begin{bmatrix} z_1^0 & z_1^1 & \dots & z_1^d \\ z_2^0 & z_2^1 & \dots & z_2^d \\ \dots & \dots & \dots & \dots \\ z_n^0 & z_n^1 & \dots & z_n^d \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_d \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix}$$

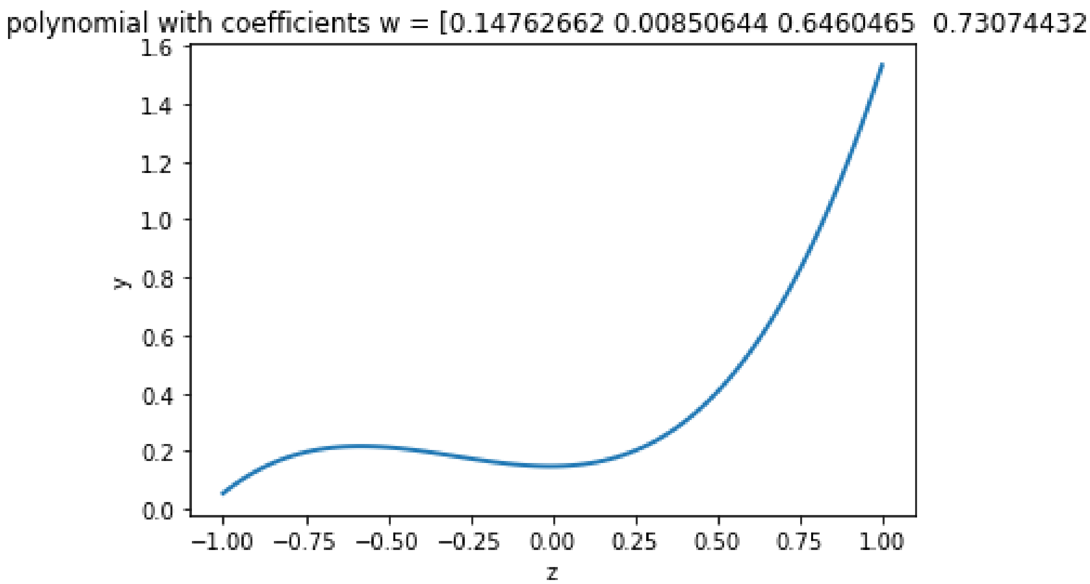
### 0.5.3 c)

[28]:
```
n = 100
z = np.linspace(-1, 1, n)
d = 3
w = np.random.rand(d + 1)
X = np.zeros((n,d + 1))

for i in range(n):
    for j in range(d + 1):
        X[i, j] = z[i]**j

p = X@w

plt.plot(z , p , linewidth=2)
plt.xlabel('z')
plt.ylabel('y')
plt.title('polynomial with coefficients w = %s' %w)
plt.show()
```


polynomial with coefficients w = [0.14762662 0.00850644 0.6460465  0.73074432]
y
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
z

```
[ ]:
```