Math ML Problem Set 1

Andi Liu, October 7th, 2022

Question 1

preliminary setup

```
import numpy as np

expenses = np.array([[2500, 2000, 2000, 2000],
                [350, 405, 325, 210],
                [200, 250, 400, 450]])

v3 = np.array([1,1,1])
v4 = np.array([1,1,1,1])
```

Verification for part b

```
# function to get monthly expenses
def get_monthly():
    return np.matmul(expenses.T, v3)

print(get_monthly()) # outputs [3050 2655 2725 2660]
```

Verification for part c

```
# function to get categorical costs over the 4 months
def get_categories():
    return np.matmul(expenses, v4)

print(get_categories()) # outputs [8500 1290 1300]
```

Verification for part d

```
# function to get total expenditure
def get_total_spending():
    return np.matmul(np.matmul(expenses, v4).T, v3)

print(get_total_spending()) # outputs 11090
```

Question 2

Preliminary setup

```
import numpy as np

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

part (a)

```
# what is a vector y so that y.T * X  is the fourth row of X?
def part_a():
    y = np.array([0, 0, 0, 1, 0])
    return np.matmul(y.T, x)

print(part_a()) # outputs [9 9 4 7]
```

part (b)

```
# given k in {1, 2, 3, 4, 5} construct a vector y st y.T * X is kth row of
X
def part_b(k):
    y = np.array([0 if i != k - 1 else 1 for i in range(1, 6)])
    return np.matmul(y.T, x)

print(part_b(1)) # outputs [8 0 1 1]
print(part_b(2)) # outputs [9 2 9 4]
print(part_b(5)) # outputs [6 9 8 9]
```

part (c)

```
# construct vector st y.T * X is a * kth row + b * jth row
def part_c(a, b, j, k):
    if j == k:
        y = np.array([0 if i != j else a + b for i in range(1, 6)])
    else:
        y = np.array([a if i == k else b if i == j else 0 for i in
range(1, 6)])
    return np.matmul(y.T, x)

print(part_c(2, 3, 1, 2)) # outputs [42 4 21 11]
```

```
print(part_c(1, 0, 0, 4)) # outputs [9 9 4 7]
print(part_c(1, 1, 4, 4)) # outputs [18 18  8 14]
```

part(d)

```
# what is a vector w s.t X * w is third column of X?
def part_d():
    w = np.array([0, 0, 1, 0])
    return np.matmul(x, w)

print(part_d()) # outputs [1 9 9 4 8]
```

part(e)

```
# construct vector w s.t X * w is kth column of X
def part_e(k):
    w = np.array([0 if i != k else 1 for i in range(1, 5)])
    return np.matmul(x, w)

print(part_e(1)) # outputs [8 9 1 9 6]
print(part_e(2)) # outputs [0 2 5 9 9]
```

part(f)

```
def part_f(a, b, j, k):
    arr = [0 for i in range(4)]
    arr[k - 1] += a
    arr[j - 1] += b
    w = np.array(arr)
    return np.matmul(x, w)

print(part_f(1, 0, 0, 1)) # outputs [8 9 1 9 6]
print(part_f(2, 3, 4, 4)) # outputs [5 20 45 35 45]
```

Question 3

```
from re import L
import numpy as np
import scipy.io as sio
import matplotlib.pyplot as plt

#collaborated with eric liu
```

```
# n = number of points
# z = points where polynomial is evaluated
# p = array to store the values of the interpolated polynomials
n = 100
z = np.linspace(-1, 1, n)
p = []


d = 3 # degree
w = np.random.rand(d)

X = np.zeros((n,d))



# generate X-matrix
for i in range(n):
    for j in range(d):
        X[i][j] = z[i] ** j

# evaluate polynomial at all points z, and store the result in p # do NOT
use a loop for this
p = np.matmul(X, w)

# plot the datapoints and the best-fit polynomials
plt.plot(z, p, linewidth=2)
plt.xlabel('z')
plt.ylabel('y')
plt.title('polynomial_with_coefficients_w_=_%s' %w )
plt.show()
```