Homework 3
ECE/CS 498 DS Spring 2019
Issued: 03/01/2019
Due: 03/08/2019
No late submission is allowed.

Name: Shuyue Lai
NetID: shuyuel2

Please submit Problem 1, 3, 6 for grading. The rest are for your practice.

Problem 1

Given the following data points in the cartesian coordinates: {(*-1, 0*), (*0, 1*), (*2, 4*)}. We are going to derive the linear regression line that fits the above samples.

a) In this sub-question, you'll derive the least square linear regression expression in terms of slope and intercept (e.g. $y = mx + b$) for the 3 given data points from first principles. Please follow the steps below.

1. Fill in the table below. An example is given for the point (-1,0).

x	y	$y' = mx + b$	$y - y'$
-1	0	$m(-1) + b = -m + b$	$0 - (-m + b) = m - b$
0	1	$m(0)+b = b$	$1-b$
2	4	$m(2)+b = 2m+b$	$4-(2m+b) = 4-2m-b$

2. Provide the expression for summation of squared error (SSE), i.e. $SSE = \sum(y - y')^2$

$SSE = \sum(y-y')^2 = (m-b)^2 + (1-b)^2 + (4-2m-b)^2$
$= m^2 - 2mb + b^2 + 1 - 2b + b^2 + 16 - 8(2m+b) + 4m^2 + 4mb + b^2$
$SSE = 5m^2 + (2b-16)m + 3b^2 - 10b + 17$

3. Simplify the expression of SSE and express it as follows:
   i. $SSE = f_1(m)$, where $f_1(m)$ is a quadratic function of the variable m. Note that terms with b are part of the co-efficient or constant.
   ii. $SSE = f_2(b)$, where $f_2(b)$ is a quadratic function of the variable b. Note that terms with m are part of the co-efficient or constant.

(i) $f_1(m) = 5m^2 + (2b-16)m + (3b^2 - 10b + 17)$

(ii) $f_2(b) = 3b^2 + (2m-10)b + (5m^2 - 16m + 17)$

Grade:_____

Homework 3
ECE/CS 498 DS Spring 2019
Issued: 03/01/2019
Due: 03/08/2019
No late submission is allowed.

Name: _____
NetID: _____

4. Find the *argmin* value for each quadratic function you derived above:
   i. Find $\underset{b}{\text{argmin}}\, f_1(m)$. The answer will be an expression for m in terms of b.
   ii. Find $\underset{m}{\text{argmin}}\, f_2(b)$. The answer will be an expression for b in terms of m.

(i) $\underset{b}{\arg\min} f_1(m)$ $f_1'(m) = 10m + 2b - 16 = 0 \Rightarrow m = \frac{16-2b}{10} = \frac{8-b}{5}$

(ii) $\underset{m}{\arg\min} f_2(m)$ $f_2'(b) = 6b + 2m - 10 = 0 \Rightarrow b = \frac{10-2m}{6} = \frac{5-m}{3}$

5. The answer from part 4 is a set of two linear equations. Solve the linear equations simultaneously for m and b. Write down the final mathematical expression of the linear regression line.

$$\begin{cases} m = \frac{8-b}{5} \\ b = \frac{5-m}{3} \end{cases} \Rightarrow 5m = 8 - \frac{5-m}{3} \Rightarrow 15m = 24 - 5 + m$$

$$\Rightarrow \begin{cases} m = \frac{19}{14} \\ b = \frac{5 - \frac{19}{14}}{3} = \frac{\frac{51}{14}}{3} = \frac{17}{14} \end{cases}$$

b) The general formula for m and b are given below:

$$m = \frac{N(\sum_{i=1}^{N} x_i y_i) - (\sum_{i=1}^{N} x_i) \cdot (\sum_{i=1}^{N} y_i)}{N(\sum_{i=1}^{N} x_i^2) - (\sum_{i=1}^{N} x_i)^2}$$

Grade:_____

$\{(-1, 0), (0, 1), (2, 4)\}$.

Homework 3 **Name:** _____
ECE/CS 498 DS Spring 2019 **NetID:** _____
Issued: 03/01/2019
Due: 03/08/2019
No late submission is allowed.

$$b = \frac{(\sum_{i=1}^{N} y_i) \cdot (\sum_{i=1}^{N} x_i^2) - (\sum_{i=1}^{N} x_i) \cdot (\sum_{i=1}^{N} x_i y_i)}{N(\sum_{i=1}^{N} x_i^2) - (\sum_{i=1}^{N} x_i)^2}$$

Where N denotes the number of data points available to fit the linear regression line. In our case, $N = 3$.

Calculate m and b by plugging the corresponding numerical values in the formulae above. Are the results the same as the value of m and b calculated in a)?

$$m = \frac{3(0+0+8)-(-1+0+2)(0+1+4)}{3(1+0+4)-(-1+0+2)^2} = \frac{24-5}{15-1} = \frac{19}{14}$$

$$b = \frac{(0+1+4)(1+0+4)-(-1+0+2)(0+0+8)}{3(1+0+4)-(-1+0+2)^2} = \frac{25-8}{15-1} = \frac{17}{14}$$

The results are the same as the value of m and b calculated in a.

Problem 2

Consider a linear regression model, where each data point is represented by input x, target variable y with the following relationship:

$$y = w \cdot x + \epsilon$$

where w is a single real-valued parameter to be learned, and ϵ, the *noise* term, is independently and identically drawn from a Gaussian distribution with mean 0 and variance 1, i.e. $\epsilon \sim N(\mu = 0, \sigma^2 = 1)$.

Provide the mathematical expression for conditional probability $p(y|w,x)$ in terms of y, w, x.

Grade:_____

Homework 3
ECE/CS 498 DS Spring 2019
Issued: 03/01/2019
Due: 03/08/2019
No late submission is allowed.

Name: _____
NetID: _____

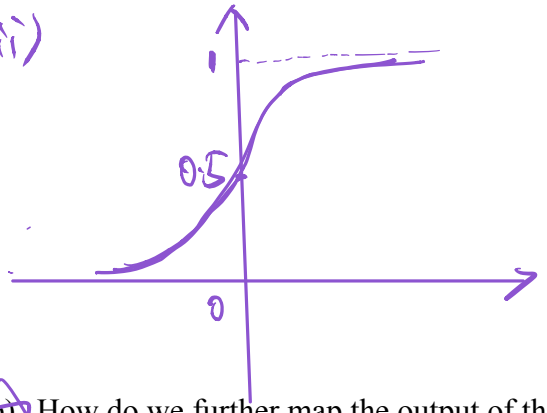
Problem 3

As we saw in lecture slides (L09 p36), in generalized linear model, $g(\mu) = \beta^T X$. Here g(.) is called an **activation function**.

a) i) Write out the expression of the activation/sigmoid function g(·) that is used in logistic regression; ii) Plot g(·);

ii) $g(\mu) = \log \frac{\mu}{1-\mu} = \beta^T x$

(ii)



logistic regression

b) How do we further map the output of the activation function to binary class labels $y \in \{-1,1\}$? [*Hint:* think in terms of *function*]

$p(x) = \mu = \frac{2}{1+e^{-\beta^T x}} - 1$

Problem 4

Draw the decision boundary and label values at both sides of the boundary for the following logistic regression classifier on the cartesian coordinate:

$$h_\theta(x) = sign(g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) - 0.5)$$

a) $\theta_0 = -10, \theta_1 = 0, \theta_2 = 7$

Grade:_____

Homework 3
ECE/CS 498 DS Spring 2019
Issued: 03/01/2019
Due: 03/08/2019
No late submission is allowed.

Name: _____
NetID: _____

b) $\theta_0 = 6, \theta_1 = -2, \theta_2 = 0$

c) $\theta_0 = 8, \theta_1 = 2, \theta_2 = 4$

Problem 5

For *binary* target variable y, compare logistic regression and Naïve Bayes. Each input variable x has *k binary* features.

a) How many parameters are needed in Naïve Bayes model?

b) How many parameters are needed in logistic regression model? (Don't forget the bias term.)

Grade:_____

Homework 3
ECE/CS 498 DS Spring 2019
Issued: 03/01/2019
Due: 03/08/2019
No late submission is allowed.

Name: _____
NetID: _____

c) Write down the conditional independence assumption in Naïve Bayes model.

d) Write down one independence assumption in the logistic regression model.

Problem 6

As described in the lecture, one way of finding the first principal component, i.e., the eigenvector corresponding to the largest eigenvalue of a matrix is to consider an arbitrary vector and keep multiplying it with the matrix till the direction of vector doesn't change any more. This algorithm is known as *power iteration* (for details, refer to: https://en.wikipedia.org/wiki/Power_iteration). In this problem we will find the largest eigenvalue and the corresponding eigenvector **using Python.**

Consider the matrix, S = $\begin{bmatrix} 10 & 3 \\ 3 & 6 \end{bmatrix}$, and x = $\begin{bmatrix} 3 \\ 5 \end{bmatrix}$. Starting with the vector x, perform power iteration to find the largest eigenvector and eigenvalue. You can follow the steps given below:

i) $x_{new} = Sx_{old}$
ii) $x_{new} = \frac{x_{new}}{||x_{new}||}$
iii) Check if the x_{new} and x_{old} are the same i.e., the algorithm has converged. If they are, then terminate. If they are not, then go back to step i) and use x_{new} as x_{old}.

Please initialize appropriately. For this problem, you can declare that the algorithm has converged if the Euclidean distance between x_{new} and x_{old} is $<10^{-5}$. Answer the following questions:

Grade:_____

Homework 3 **Name:** _____
ECE/CS 498 DS Spring 2019 **NetID:** _____
Issued: 03/01/2019
Due: 03/08/2019
No late submission is allowed.

1. How many iterations of the above algorithm are required to converge to the first principal component?

   11 times

2. What is the first principal components i.e., eigenvector corresponding to the largest eigenvalue?

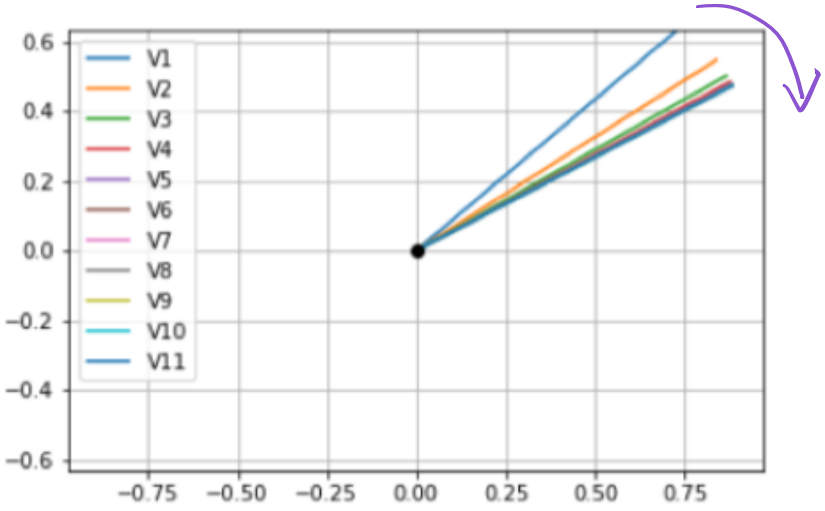
   $V = [0.88167215, 0.47186251]$

3. What is the largest eigenvalue?

   $\begin{bmatrix} 10 & 3 \\ 3 & 6 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = \begin{vmatrix} 10-\lambda & 3 \\ 3 & 6-\lambda \end{vmatrix} = 60 - 16\lambda + \lambda^2 - 9 = 0 \Rightarrow \lambda^2 - 16\lambda + 51$

   $(\lambda - 8)^2 - 13 \Rightarrow \lambda = \pm\sqrt{13} + 8 \Rightarrow \lambda = \sqrt{13} + 8 = 11.60$

4. On the same plot, plot the vectors from all iterations. What do you observe? (Hint: Make sure to normalize each vector to be unit length for better visualization).



Vectors turns out to converge to the final result.

(See code below)

Grade:_____

```
#!/usr/bin/python

import numpy as np
import scipy.spatial

def power_iteration(b_k, A):
    # Ideally choose a random vector
    # To decrease the chance that our vector
    # Is orthogonal to the eigenvector
    # b_k = np.random.rand(A.shape[1])
    count = 0
    b_ks = []

    while(True):
        count += 1

        # calculate the matrix-by-vector product Ab
        b_k1 = np.dot(A, b_k)

        # calculate the norm
        b_k1_norm = np.linalg.norm(b_k1)

        # re normalize the vector
        b_k1 = b_k1 / b_k1_norm

        if(scipy.spatial.distance.euclidean(b_k, b_k1) < 0.00001):
            b_k = b_k1
            break
        else:
            b_ks.append(b_k1)
            b_k = b_k1

    return b_k, count, b_ks

b_k, count, b_ks = power_iteration(np.array([3, 5]), np.array([[10, 3], [3, 6]]))

print(b_k)
print(count)
```

```
[0.88167215 0.47186251]
12
```

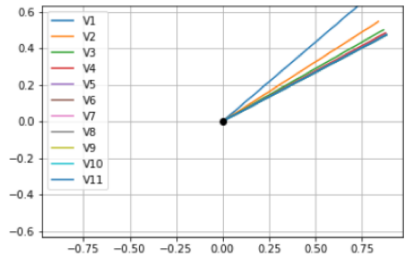
```
import numpy as np
import matplotlib.pyplot as plt
M = np.array(b_ks)

rows,cols = M.T.shape

#Get absolute maxes for axis ranges to center origin
#This is optional
maxes = 1.1*np.amax(abs(M), axis = 0)

for i,l in enumerate(range(0,cols)):
    xs = [0,M[i,0]]
    ys = [0,M[i,1]]
    plt.plot(xs,ys)

plt.plot(0,0,'ok') #<-- plot a black point at the origin
plt.axis('equal')  #<-- set the axes to the same scale
plt.xlim([-maxes[0],maxes[0]]) #<-- set the x axis limits
plt.ylim([-maxes[1],maxes[1]]) #<-- set the y axis limits
plt.legend(['V'+str(i+1) for i in range(cols)]) #<-- give a legend
plt.grid(b=True, which='major') #<-- plot grid lines
plt.show()
```



```
import numpy as np
import matplotlib.pyplot as plt
M = np.array(b_ks)

rows,cols = M.T.shape

#Get absolute maxes for axis ranges to center origin
#This is optional
maxes = 1.1*np.amax(abs(M), axis = 0)

for i,l in enumerate(range(0,cols)):
    xs = [0,M[i,0]]
    ys = [0,M[i,1]]
    plt.plot(xs,ys)

plt.plot(0,0,'ok') #<-- plot a black point at the origin
plt.axis('equal')  #<-- set the axes to the same scale
plt.xlim([-maxes[0],maxes[0]]) #<-- set the x axis limits
plt.ylim([-maxes[1],maxes[1]]) #<-- set the y axis limits
plt.legend(['V'+str(i+1) for i in range(cols)]) #<-- give a legend
plt.grid(b=True, which='major') #<-- plot grid lines
plt.show()
```