ESE 542-HW5
wjeong@seas.upenn.edu

Problem 1.

(a)

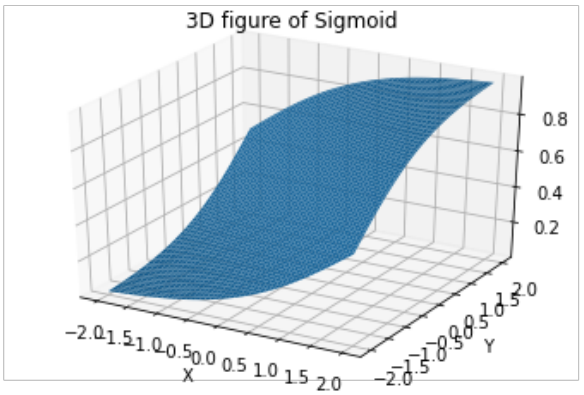


Figure1. Plot of sigmoid

(b)

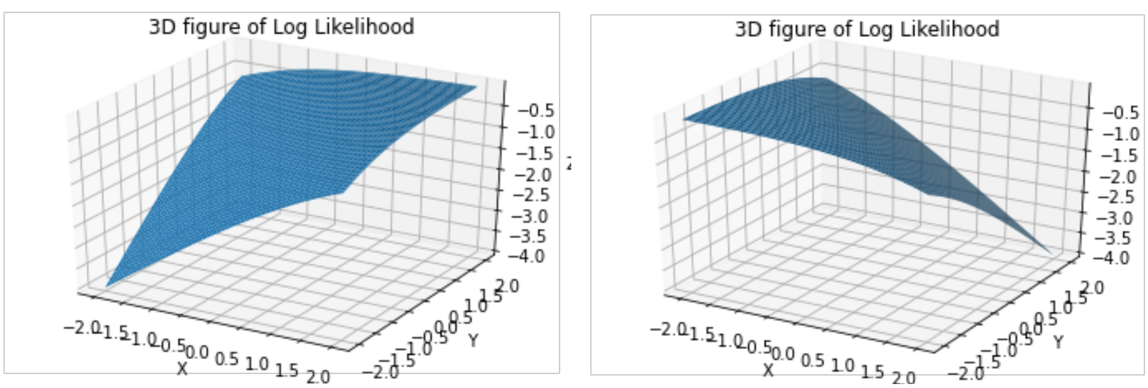


Figure2. Plot of Log Likelihood

*Yes, it is possible to maximize the function because the function is monotonic.

ESE 542 - HW5

Wontae Jeong (wjeong@seas.upenn.edu)

Problem 1.

(b) $Y \in \{-1, 1\}$ $\Rightarrow P(Y|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$

If we set $\begin{cases} P(Y=+1 \mid X=x) = p(x) \\ P(Y=-1 \mid X=x) = 1-p(x) \end{cases}$

We know $\begin{cases} p(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}} \\ 1-p(x) = 1 - \frac{1}{1+e^{-(\beta_0+\beta_1 x)}} \end{cases}$

$$\cdot\, 1-p(x) = 1 - \frac{1}{1+e^{-(\beta_0+\beta_1 x)}} = \frac{(1+e^{-(\beta_0+\beta_1 x)})-1}{1+e^{-(\beta_0+\beta_1 x)}}$$

$$\Rightarrow \frac{e^{-(\beta_0+\beta_1 x)}}{1+e^{-(\beta_0+\beta_1 x)}} \times \frac{e^{+(\beta_0+\beta_1 x)}}{e^{+(\beta_0+\beta_1 x)}}$$

$$\Rightarrow \frac{1}{1+e^{(\beta_0+\beta_1 x)}}$$

$$\therefore \begin{cases} P(Y=+1 \mid X=x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}} \\ P(Y=-1 \mid X=x) = \frac{1}{1+e^{(\beta_0+\beta_1 x)}} \end{cases}$$

Thus, if $Y \in \{-1, 1\}$, the probability of Y given X can be written as

$$P(Y|X) = \frac{1}{1+e^{-y(\beta_0+\beta_1 x)}}$$

The log likelihood function for m data points,

$$\mathcal{L} = \prod_{i=1}^{m} \frac{1}{1+e^{-y_i(\beta_0+\beta_1 x)}}$$

$$\Rightarrow \ln \mathcal{L} = \ln \prod_{i=1}^{m} \frac{1}{1+e^{-y_i(\beta_0+\beta_1 x)}}$$

$$= -\sum_{i=1}^{m} \ln\left(1+e^{-y_i(\beta_0+\beta_1 x)}\right)$$

Problem2

• Why is random splitting better than sequential splitting in our case?

In our dataset, only 0 and 1 are used and arranged in order. Thus, if we sequentiall split the dataset, then there will be no points for the classifier because the only one digit of dataset will be tested during test. We should use random splitting.

• Derive the classification rule for the threshold 0.5

We classify as +1 if P(Y = 1|X) ≥ 0.5.

$$\bullet P(Y = +1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta x^T)}} \geq 0.5$$

$$\frac{1}{1 + e^{-(\beta_0 + \beta x^T)}} \geq 0.5$$

$$e^{-(\beta_0 + \beta x^T)} \leq 1$$

$$\beta_0 + \beta x^T \geq 0$$

$$\bullet P(Y = -1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta x^T)}} \leq 0.5$$

$$\frac{1}{1 + e^{-(\beta_0 + \beta x^T)}} \leq 0.5$$

$$e^{-(\beta_0 + \beta x^T)} \geq 1$$

$$\beta_0 + \beta x^T \leq 0$$

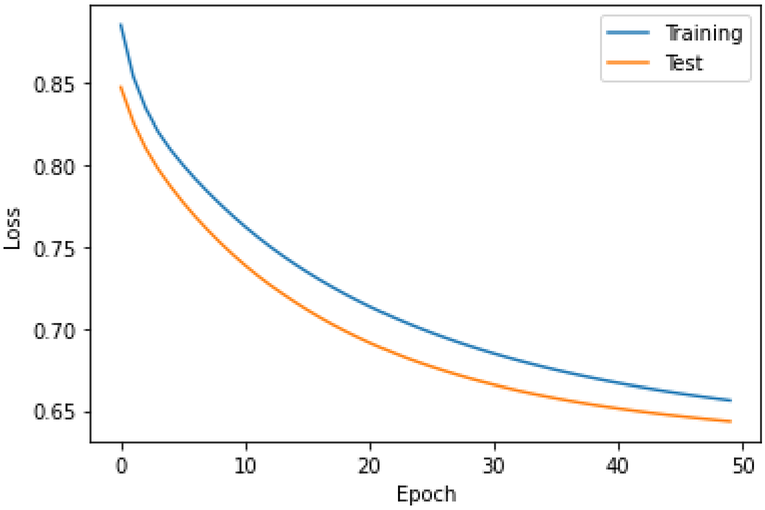
Thus, P(Y|X) is a **sign** function of $(\beta_0 + \beta x^T)$

```
import torchvision.transforms as transforms
import torch.optim as optim
import matplotlib.pyplot as plt
import numpy as np
import cv2
from sklearn.model_selection import train_test_split
import pdb
```

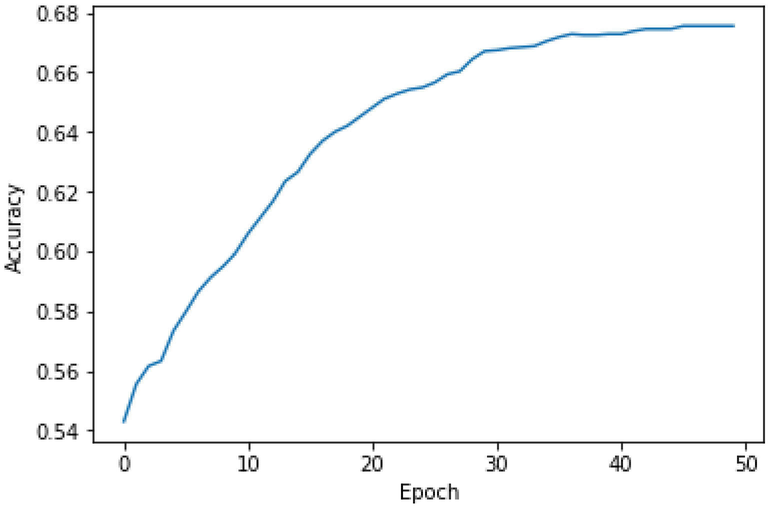
```
import pdb

def compute_loss(data, labels, B, B_0):
    t_power = -labels*(np.dot(data,B.T)+B_0)
    t_exp = np.exp(t_power)
    logloss = np.mean(np.log(1 + t_exp))
    return logloss

def compute_gradients(data, labels, B, B_0):
    dB = 0
    dB_0 = 0
    t_power = -labels*(np.dot(data,B.T)+B_0)
    t_exp = np.exp(t_power)
    dB = -np.dot(data.T,(labels*t_exp/(1 + t_exp)))//data.shape[0]
    temp = t_exp/(1 + t_exp)
    dB_0 = -np.dot(labels.T, temp)//data.shape[0]
    return dB, dB_0

if __name__ == '__main__':
    x = np.load('/content/drive/MyDrive/Colab Notebooks/ESE542/hw5/data.npy')
    y = np.load('/content/drive/MyDrive/Colab Notebooks/ESE542/hw5/label.npy')
    accuracy_test = []
    loss_test = []
    error_test = []
    loss_train = []
    y[y== 1] = -1
    y[y== 0] = 1
    y = y.reshape(-1, 1)
    x = x//255.0
    B = np.random.randn(1, x.shape[1])
    B_0 = np.random.randn(1)
    print(x)
    x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2,random_
    lr = 0.05
    total = 0.0
    correct = 0
    error = 0
    epoch = 0

    for _ in range(50):
        ## Compute Loss
        loss = compute_loss(x_train, y_train, B, B_0) ## Compute Gradient
        dB, dB_0 = compute_gradients(x_train, y_train, B, B_0)

        ## Update Parameters
        B = B - lr*dB.T
        B_0 = B_0 - lr*dB_0

        ##Compute Accuracy and Loss on Test set (x_test, y_test)
        outputs = np.dot(x_test,B.T)+B_0
        outputs[outputs<0] = -1
        outputs[outputs>0] = 1
        loss_train.append(loss)
        if _ % 10 == 0:
            print('Epoch [{}/{}], Train Loss: {:.4f}'.format(_+1, 50,loss))
        total = y_test.shape[0]
```

```
        total = y_test.shape[0]
        t_loss = compute_loss(x_test, y_test, B, B_0)

        correct = (outputs == y_test).sum().item()
        if _ % 10 == 0:
            print('Epoch [{}/{}], Test Loss: {:.4f}'.format(_+1, 50,t_loss))
        accuracy = 100 * correct / total
        error = 100 - accuracy
        accuracy_test.append(accuracy/100.0)
        loss_test.append(t_loss)
        error_test.append(error/100.0)
        ##Plot Loss and Accuracy
    plt.plot(loss_train)
    plt.plot(loss_test)
    plt.legend(["Training", "Test"])
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
Epoch [1/50], Train Loss: 0.8856
Epoch [1/50], Test Loss: 0.8475
Epoch [11/50], Train Loss: 0.7625
Epoch [11/50], Test Loss: 0.7388
Epoch [21/50], Train Loss: 0.7136
Epoch [21/50], Test Loss: 0.6914
Epoch [31/50], Train Loss: 0.6848
Epoch [31/50], Test Loss: 0.6657
Epoch [41/50], Train Loss: 0.6668
Epoch [41/50], Test Loss: 0.6514
```




```
plt.plot(accuracy_test)
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.show()
```



0s completed at 7:19 PM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.

Problem 3 (OH)

1. $P(Y=y \mid X=x)$

i) $P(y=0) = P(y=1) = 0.5$

ii) the classification accuracy of any classifier is at most 0.9.

iii) the accuracy of Bayes optimal possible classifier is at least 0.8.

$x \sim \text{Unif}[0,1]$

- $P(Y=1 \mid X=x) = \begin{cases} p & x \geq 0.5 \\ q, & \text{o.w} \end{cases}$ $\underline{\cdot\ p+q=1}$

- $P(Y=0 \mid X=x) = \begin{cases} 1-p & x \geq 0.5 \\ 1-q & \text{o.w} \end{cases}$

probability given x should add to 1.

Lec 16.

Let $h(x)$ is a function of classifier.

If $h(x)=1$, then it's correct $p\%$ of the time.

then the error for the classification.

$= P_r\{h(x) \neq y\}$ $\{y=1\}$

$= E\{\mathbb{1}_{\{h(x)\neq y\}}\}$

$= E\{P(Y=1 \mid X=x)\mathbb{1}_{\{h(x)\neq 1\}} + P(Y=0 \mid X=x)\mathbb{1}_{\{h(x)\neq 0\}}\}$

$q \cdot 0.5 + (1-p)\cdot 0.5$

— If the accuracy is at most 0.9

$0.5q + 0.5(1-p) \geq 0.1$

- If the Bayes optimal possible classifier is at least 0.8

$$0.5q + 0.5(1-p) \leq 0.2$$

Thus. $\begin{cases} \cdot\ 0.5q + 0.5(1-p) \geq 0.1 \\ \cdot\ 0.5q + 0.5(1-p) \leq 0.2 \\ \cdot\ p+q=1 \end{cases}$

$$0.1 \leq 0.5q + 0.5(1-p) \leq 0.2$$

$$\Rightarrow 0.2 \leq q+1-p \leq 0.4$$

$$\Rightarrow -0.4 \leq p-q-1 \leq -0.2$$

$$\Rightarrow 0.6 \leq p-q \leq 0.8$$

$$\cdot\ \underline{p+q=1}$$

$$\Rightarrow 0.6 \leq 2p-1 \leq 0.8$$

$$\boxed{\begin{cases} 0.8 \leq p \leq 0.9 \\ 0.1 \leq q \leq 0.2 \end{cases}}$$

I picked $p=0.8$, $q=0.2$.

Problem3

Based on the design, I picked p = 0.8 and q = 0.2

```
import matplotlib.pyplot as plt
import numpy as np
import cv2
from sklearn.model_selection import train_test_split
import pdb
from sklearn.datasets import load_digits
```

2. Binary Classifier using Logistic Regression

```
x_train= np.random.uniform(0,1,100)
x_test = np.random.uniform(0,1,100)
y_train = np.zeros(100)
y_test = np.zeros(100)
for i in range(100):
  if x_train[i] >=0.5:
    y_train[i] = np.random.binomial(1,0.80,1)
  else:
    y_train[i] = np.random.binomial(1,0.20,1)
for i in range(100):
  if x_test[i] >=0.5:
    y_test[i] = np.random.binomial(1,0.80,1)
  else:
    y_test[i] = np.random.binomial(1,0.20,1)
```

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()
clf.fit(x_train.reshape(-1,1), y_train)
print('The accuracy score=', clf.score(x_test.reshape(-1,1),y_test))
```

```
The accuracy score= 0.76
```

3. Bayes optimal classifier

```
predict = np.zeros(100)
for i in range(100):
  if x_test[i] >= 0.5:
    predict[i] = 1
  else:
    predict[i] = 0
total  = 0
```

```
for i in range(100):
  if predict[i]==y_test[i]:
    total +=1
  else:
    total +=0
accuracy = total/100

print('The accuracy score=', accuracy)
```

```
The accuracy score= 0.81
```

4. n=1000

```
x_train= np.random.uniform(0,1,1000)
x_test = np.random.uniform(0,1,1000)
y_train = np.zeros(1000)
y_test = np.zeros(1000)
for i in range(1000):
  if x_train[i] >=0.5:
    y_train[i] = np.random.binomial(1,0.80,1)
  else:
    y_train[i] = np.random.binomial(1,0.20,1)
for i in range(1000):
  if x_test[i] >=0.5:
    y_test[i] = np.random.binomial(1,0.80,1)
  else:
    y_test[i] = np.random.binomial(1,0.20,1)
```

```
clf = LogisticRegression()
clf.fit(x_train.reshape(-1,1), y_train)
print('The accuracy score=',clf.score(x_test.reshape(-1,1),y_test))
```

```
The accuracy score= 0.819
```

```
predict = np.zeros(1000)
for i in range(1000):
  if x_test[i] >= 0.5:
    predict[i] = 1
  else:
    predict[i] = 0

total  = 0
for i in range(1000):
  if predict[i]==y_test[i]:
    total +=1
  else:
```

```
    total +=0

print('The accuracy score=',accuracy)
```

```
The accuracy score= 0.821
```

- Accuracy

- Binary Classifier(n = 100) = 0.76
- Bayes Optimal Classifier(n = 100) = 0.81
- Binary Classifier(n = 1000) = 0.819
- Bayes Optimal Classifier(n = 1000) = 0.821

Thus, we can see that the classifier where n = 1000 has better accuracy than where n = 100. Also, the Bayes optimal classifier has better accuracy than the binary classifier using logistic regression.

✓ 0s completed at 9:44 PM

Problem 4

1. $$\sum_{k=1}^{K} \sum_{x \in C_k} \sum_{i=1}^{p} (c_{ki} - x_i)^2$$

c_k of the k-th cluster C_k and the data x contains p features

$$\sum_{k=1}^{K} \sum_{x \in C_k} \sum_{i=1}^{p} (c_{ki} - x_i)^2$$

take partial derivative by c_{ki} $\Rightarrow$ $$\frac{\partial}{\partial c_{ki}} \sum_{k=1}^{K} \sum_{x \in C_k} \sum_{i=1}^{p} (c_{ki} - x_i)^2$$

$$\Rightarrow \cancel{\sum_{k=1}^{K}} \sum_{x \in C_k} \cancel{\sum_{i=1}^{p}} 2(c_{ki} - x_i)$$

$$\sum_{x \in C_k} 2(c_{ki} - x_i) = 0$$

$$\Rightarrow \sum_{x \in C_k} c_{ki} = \sum_{x \in C_{ki}} x_i$$

$$c_{ki} \cdot \sum_{x \in C_k} 1 = \sum_{x \in C_{ki}} x_i$$

$$c_k \cdot n = \sum_{x \in C_k} x_i \quad \left(n = \sum_{x \in C_k} 1, \ n \text{ is number of samples in kth cluster} \right)$$

$$c_k = \frac{1}{n} \sum_{x \in C_i} x_i$$

$\therefore$ Thus c_k is the mean of the points in the cluster.

2. $$\sum_{k=1}^{K} \sum_{x \in C_k} \sum_{i=1}^{p} |c_{ki} - x_i|$$

If taking partial derivative by c_{ki},

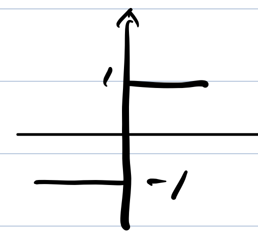
$$\sum_{k=1}^{K} \sum_{x \in C_k} \sum_{i=1}^{p} |c_{ki} - x_i|$$

$$\Rightarrow \frac{\partial}{\partial c_{k_i}} \sum_{k=1}^{K} \sum_{x \in C_k} \sum_{i=1}^{l} |c_{k_i} - x_i|$$

$$\Rightarrow \frac{\partial}{\partial c_{k_i}} \sum_{x \in C_k} |c_{k_i} - x_i|$$

$$\Rightarrow \sum_{x \in C_k} \operatorname{sgn}(c_{k_i} - x_i)$$

$$\begin{cases} c_{k_i} - x_i > 0. \Rightarrow 1 \\ c_{k_i} - x_i < 0 \Rightarrow -1. \end{cases}$$



Because, the number of $|\{x_i \mid x_i < c_{k_i}\}|$ is equal to $|\{x_i \mid x_i > c_{k_i}\}|$

Thus, this is the median of the points in the cluster.