Lewis Arnsten

Homework 3

1. a) for any $w \neq \hat{w}$ and $r := y - Xw$ show $r = \hat{r} - X(\hat{w} - w)$

$$r - \hat{r} = y - Xw - (y - X\hat{w})$$
$$r - \hat{r} = -Xw + X\hat{w}$$
$$r - \hat{r} = X(\hat{w} - w)$$
$$r = \hat{r} + X(\hat{w} - w)$$

b) show $\|r^2\| = \hat{r}^T\hat{r} + \hat{r}^T X(\hat{w}-w) + (\hat{w}-w)^T X^T \hat{r} + (\hat{w}-w)^T X^T X(\hat{w}-w)$

$$\|r^2\| = \|\hat{r} + X(\hat{w}-w)\|^2$$
$$= (\hat{r} + X(\hat{w}-w))^T(\hat{r} + X(\hat{w}-w))$$
$$= \hat{r}^T\hat{r} + \hat{r}^T X(\hat{w}-w) + (\hat{w}-w)^T X^T \hat{r} + (\hat{w}-w)^T X^T X(\hat{w}-w)$$

c) $\hat{r}$ is orthogonal to columns of $X \rightarrow \hat{r}^T X = 0$

expression above $= \hat{r}^T\hat{r} + (\hat{w}-w)^T X^T X(\hat{w}-w)$

d) Since X^TX is positive definite $\therefore (\hat{w}-w)^T X^T X(\hat{w}-w) > 0$

Thus $\|r^2\| = r^Tr = \hat{r}^T\hat{r} + (\hat{w}-w)^T X^T X(\hat{w}-w) > \hat{r}^T\hat{r}$

$= \|\hat{r}\|^2 \longrightarrow \|r\| > \|\hat{r}\|$ and $\hat{w}$ is the least squares solution

2. a) $f(w) = w^T(2x)$

$$\nabla_w f = 2x$$

b) $f(w) = 6w^T x - 1.5 x^T w$

$$\nabla_w f = 4.5x$$

c) $f(w) = w^T \begin{bmatrix} 12 & 7 \\ 7 & 3 \end{bmatrix} w$ $\quad k=i=j$

$$\nabla_w f = 2\begin{bmatrix} 12 & 7 \\ 7 & 3 \end{bmatrix} w = \begin{bmatrix} 24 & 14 \\ 14 & 6 \end{bmatrix} w$$

d) $f(w) = w^T \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix} w$

$$\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 2w_1 + 4w_2 \\ 6w_1 + 8w_2 \end{bmatrix} [w_1 \; w_2]$$

$$= 2w_1^2 + 4w_1w_2 + 6w_1w_2 + 8w_2^2$$

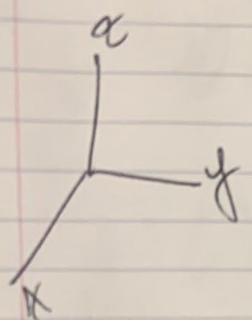
$$= 2w_1^2 + 10w_1w_2 + 8w_2^2$$

$$\nabla_w f = (4w_1 + 10w_2, \; 10w_1 + 16w_2)$$

$$= \begin{bmatrix} 4 & 10 \\ 10 & 16 \end{bmatrix} w$$

3. $x = \begin{bmatrix} 3 & 1 \\ 0 & 3 \\ 0 & 4 \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}$

a) $u_1 = \frac{x_1}{\|x_1\|_2} = \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} / 3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$



$x_2' = x_2 - u_1 u_1^T x_2$

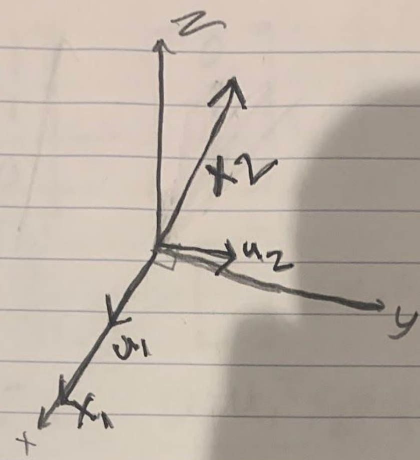
$u_1 u_1^T x_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

$x_2' = x_2 - \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 4 \end{bmatrix}$

$u_2 = \frac{x_2'}{\|x_2'\|} = \begin{bmatrix} 0 \\ 3 \\ 4 \end{bmatrix} / 5 = \begin{bmatrix} 0 \\ 3/5 \\ 4/5 \end{bmatrix}$

$U = \begin{bmatrix} 1 & 0 \\ 0 & 3/5 \\ 0 & 4/5 \end{bmatrix}$

3. b)



$$X = \begin{bmatrix} 3 & 1 \\ 0 & 3 \\ 0 & 4 \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 3/5 \\ 0 & 4/5 \end{bmatrix}$$

c) $w = \arg\min \|y - Xw\|_2^2 = (X^TX)^{-1}X^Ty$

$\tilde{w} = \arg\min \|y - Uw\|_2^2 = (U^TU)^{-1}U^Ty = U^Ty$

$\hat{y} = Xw = X(X^TX)^{-1}X^Ty$

or $\hat{y} = U\tilde{w} = UU^Ty$

$$\hat{y} = \begin{bmatrix} 1 & 0 \\ 0 & 3/5 \\ 0 & 4/5 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/4 & 4/5 \end{bmatrix}\begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 39/25 \\ 52/25 \end{bmatrix}$$

4, a) $x_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$ $x_2 = \begin{bmatrix} 0 \\ -2 \\ 1 \end{bmatrix}$ $x_3 = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$ on $\{x \in \mathbb{R}^3 : x_1 - x_2 - 2x_3 = 0\}$

$$u_1 = \frac{x_1}{\|x_1\|_2} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} / \sqrt{2} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix}$$

$$x_2' = x_2 - u_1 u_1^T x_2 = x_2 - \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

$$u_2 = \frac{x_2'}{\|x_2'\|} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} / \sqrt{3} = \begin{bmatrix} 1/\sqrt{3} \\ -1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix}$$

$$x_3' = x_3 - u_1 u^T x_3 - u_2 u_2 x_3 = 0$$

$$\text{Proj}_V\ P = u_1 u_1^T + u_2 u_2^T = \begin{bmatrix} 5/6 & 1/6 & 1/3 \\ 1/6 & 5/6 & -1/3 \\ 1/3 & -1/3 & 1/3 \end{bmatrix} = P$$

b) The matrix rank is 2

c) $$Px - x = \begin{pmatrix} 4/3 \\ 2/3 \\ 1/3 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1/3 \\ -1/3 \\ -2/3 \end{pmatrix} \rightarrow \sqrt{(1/3)^2 + (-1/3)^2 + (-2/3)^2}$$

$$\text{distance} = \sqrt{2/3}$$

```python
import numpy as np
from scipy.io import loadmat
import matplotlib.pyplot as plt
from random import randrange

#5)
def GS(matrix):
    transposed = np.hstack([np.array([c]).T for c in matrix])
    without_zeros = []
    for row in transposed:
        if not all([c == 0 for c in row]):
            without_zeros.append(row)

    norm = np.linalg.norm(without_zeros[0])
    u1 = without_zeros[0] / norm
    Us = [u1]
    Xjs = [u1]
    for c in range(2, len(without_zeros) + 1):
        Xi = without_zeros[c-1:c][0]
        norm = np.linalg.norm(Xi)
        XiN = Xi / norm
        Us.append(XiN)

        FoundU = []
        for i in range(0,len(Us)-1):
            u2 = Us[i].dot(Us[i].T).dot(Xi)
            FoundU.append(u2)

        Xj_prime = without_zeros[c - 1] - sum(FoundU)

        if not all([x == 0 for x in Xj_prime]):
            Xjs.append(Xj_prime / np.linalg.norm(Xj_prime))

    return np.hstack(Xjs)

x2 = [[3,1],[0,3],[0,4]]
mx = np.matrix(x2)
print(GS(mx))
```

```
lewiss-mbp:desktop lewisarnsten$ python3 PSet3_LA.py
[[1.   0. ]
 [0.   0.6]
 [0.   0.8]]
```

```python
#6)
#a. Since the flower dataset contains full words as labels, we need to change the labels
# into a form that can be used with least squares. I assigned a whole number between
# 1 and 3 to each of the three flower names. This produces three distinct catagories.
# Thus, for a new array of feaures, the least squares weights will produce a result between ~0-3
# that can be  turned into a prediction by assigning it to the closes whole number.
data = loadmat('fisheriris.mat')
flower_x = data['meas']
flower_y = data['species']

flower_labels = []
for f in flower_y:
    if f == "virginica": #100-150
        flower_labels.append([3])
    elif f == "versicolor": #50-100
        flower_labels.append([2])
    elif f == "setosa": #0-50
        flower_labels.append([1])
flower_w = np.linalg.inv(flower_x.T@flower_x)@flower_x.T@flower_labels
```

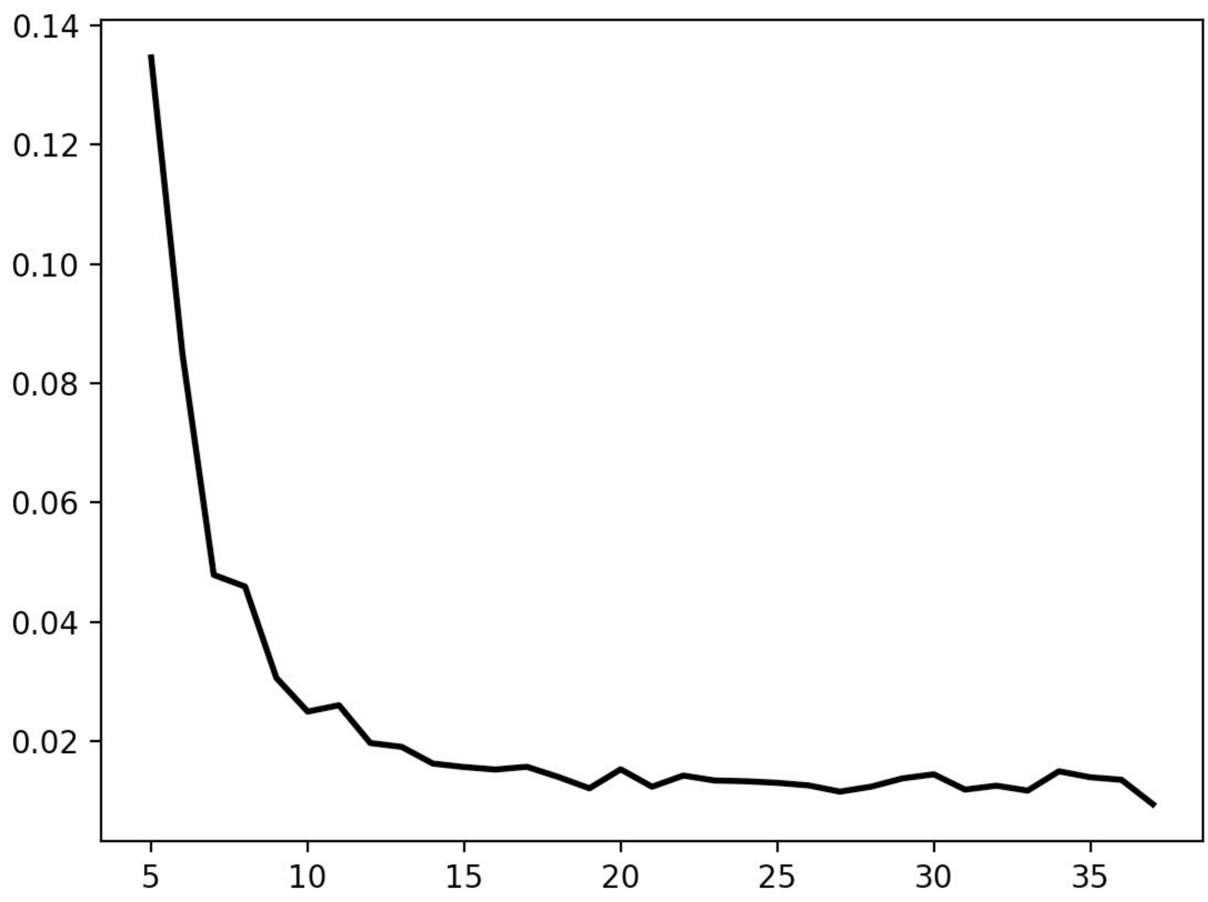
```python
#b.
def LS(set_size):
    y_testlabels = [[],[],[]]
    y_trainlabels = [[],[],[]]
    x_trainfeatures = [[],[],[]]
    x_testfeatures = [[],[],[]]
    used_nums = []
    new_num = []
    for i in range(int(set_size)):
        ran_num1 = randrange(50)
        while ran_num1 in used_nums:
            ran_num1 = randrange(50)
        ran_num2 = 50 + ran_num1
        ran_num3 = 100 + ran_num1
        y_trainlabels[0].append(flower_labels[ran_num1])
        y_trainlabels[1].append(flower_labels[ran_num2])
        y_trainlabels[2].append(flower_labels[ran_num3])
        x_trainfeatures[0].append(flower_x[ran_num1])
        x_trainfeatures[1].append(flower_x[ran_num2])
        x_trainfeatures[2].append(flower_x[ran_num3])
        '''
        y_trainlabels[0].append(flower_labels[ran_num1][0:3])
        y_trainlabels[1].append(flower_labels[ran_num2][0:3])
        y_trainlabels[2].append(flower_labels[ran_num3][0:3])
        x_trainfeatures[0].append(flower_x[ran_num1][0:3])
        x_trainfeatures[1].append(flower_x[ran_num2][0:3])
        x_trainfeatures[2].append(flower_x[ran_num3][0:3])
        '''
        used_nums.append(ran_num1)
        used_nums.append(ran_num2)
        used_nums.append(ran_num3)
```

```python
for i in range(0,150):
    if i not in used_nums:
        new_num.append(i)
        if i < 50:
            y_testlabels[0].append(flower_labels[i])
            x_testfeatures[0].append(flower_x[i])
            #y_testlabels[0].append(flower_labels[i][0:3])
            #x_testfeatures[0].append(flower_x[i][0:3])
        if 50 <= i  and i < 100:
            y_testlabels[1].append(flower_labels[i])
            x_testfeatures[1].append(flower_x[i])
            #y_testlabels[1].append(flower_labels[i][0:3])
            #x_testfeatures[1].append(flower_x[i][0:3])
        if 100 <= i and i < 150:
            y_testlabels[2].append(flower_labels[i])
            x_testfeatures[2].append(flower_x[i])
            #y_testlabels[2].append(flower_labels[i][0:3])
            #x_testfeatures[2].append(flower_x[i][0:3])

flower_w1 = np.linalg.inv(np.matrix(x_trainfeatures[0]).T@np.matrix(x_trainfeatures[0]))@np.matrix(x_trainfeatures[0]).T@np.matrix(y_trainlabels[0])
flower_w2 = np.linalg.inv(np.matrix(x_trainfeatures[1]).T@np.matrix(x_trainfeatures[1]))@np.matrix(x_trainfeatures[1]).T@np.matrix(y_trainlabels[1])
flower_w3 = np.linalg.inv(np.matrix(x_trainfeatures[2]).T@np.matrix(x_trainfeatures[2]))@np.matrix(x_trainfeatures[2]).T@np.matrix(y_trainlabels[2])
```

```python
    y_hat1 = np.matrix(x_testfeatures[0]) @ flower_w1
    for i in y_hat1:
        if i[0] < 1.5:
            i[0] = 1
    mistakes1 = np.sum(y_hat1 != y_testlabels[0])
    y_hat2 = np.matrix(x_testfeatures[1]) @ flower_w2

    for i in y_hat2:
        if 1.5 <= i[0] and i[0] < 2.5:
            i[0] = 2
    mistakes2 = np.sum(y_hat2 != y_testlabels[1])

    y_hat3 = np.matrix(x_testfeatures[2]) @ flower_w3
    for i in y_hat3:
        if 2.5 <= i[0]:
            i[0] = 3
    mistakes3 = np.sum(y_hat3 != y_testlabels[2])

    error = (mistakes1 + mistakes2 + mistakes3) / (len(y_hat1)*3)

    return error

errors = 0
for i in range(10):
    errors = errors + LS(40)
print(errors/10)
```

```
0.016666666666666666
```

```
#c.
error_per_ss = []
set_size = []
for i in range(5,38):
    avg_error = 0
    for n in range(100):
        avg_error = avg_error + LS(i)
    final_error = avg_error /100
    set_size.append(i)
    error_per_ss.append(final_error)

plt.plot(set_size, error_per_ss, linewidth=2, color='black')
plt.show()
```



D. The code to redesign the classifier for only three elements is commented in the code for part a above. To find the average error I simply uncommented that code (for both testing and training) and commented the related lines immediately above. The average error was:

```
0.026666666666666665
```