Question 2

In [ ]:

```
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt
import scipy.linalg as sla
from mpl_toolkits.mplot3d import Axes3D
import numpy.linalg as la
import math

import os
os.chdir(r"D:\Documents\Harris\Padhai\4. Quarter 4\Math for ML\Homeworks\Homework 2")
```

In [11]:

```
#loading data matrix X and Y
d_face = sio.loadmat('face_emotion_data.mat', squeeze_me = True)
X = d_face['X']
y = d_face['y']
```

In [12]:

```
from sklearn.model_selection import KFold
kf8 = KFold(n_splits=8, shuffle=False)
kf7 = KFold(n_splits=7, shuffle=False)
```

In [13]:

```
def error_func(w, Xtest, Ytest):

    yhat = Xtest@w

    error = []
    for i in range(len(Ytest)):
        error.append(np.subtract(Ytest, yhat))

    return sum(error)/len(error)
```

Question 2a.

In [28]:

```
def truncatedSVD(X_data, y_data):
    errors = []

    import math
    for train_index_B, test_index_B in kf8.split(X_data):
        x_trainB = X_data[train_index_B]
        x_holdoutB = X_data[test_index_B]

        y_trainB = y_data[train_index_B]
        y_holdoutB = y_data[test_index_B]

        for train_index, test_index in kf7.split(x_trainB):
            x_train = x_trainB[train_index]
            x_test = x_trainB[test_index]

            y_train = y_trainB[train_index]
            y_test = y_trainB[test_index]

            U, S, V = la.svd(x_train,full_matrices=False)
            S_inv = 1/S
            S_x = np.diag(S_inv)
            S_x[:,1:] = 0

            min_w = V@S_x@U.T@y_train
            #print(min_w)

            min_error = error_func(min_w, x_test, y_test)
            #print(min_error.shape)

            for i in range(2,10):

                Ut, St, Vt = la.svd(x_train,full_matrices=False)
                S_invt = 1/St
                S_xt = np.diag(S_invt)
                S_xt[:,i:] = 0
                w = V@S_xt@U.T@y_train
                #print(w.shape)

                error = error_func(w, x_test, y_test)
                #print(error.shape)
                if((sum(error)/len(error))<(sum(min_error)/len(error))):
                    min_w = w
                    min_error = error


            error_holdout = error_func(min_w, x_holdoutB, y_holdoutB)
            errors.append(error_holdout)
            #print(errors)

    return errors
```

In [66]:

```
avgerror1 = truncatedSVD(X, y)
print(len(avgerror1))
print((sum(avgerror1)/len(avgerror1))/16)
```

```
56
[-0.00583885 -0.02032973  0.02304806 -0.0296349   0.043372   -0.01156517
  0.00634074  0.00404593 -0.00653409  0.06171246 -0.02414963  0.04552239
  0.00570181  0.02030202 -0.00534707  0.00451263]
```

In [68]:

```
list1 = [-0.00583885, -0.02032973,  0.02304806, -0.0296349,   0.043372,   -0.01156517,
0.00634074, 0.00404593, -0.00653409,
         0.06171246, -0.02414963,  0.04552239, 0.00570181,  0.02030202, -0.00534707,
0.00451263]
print('The average error is:', sum(list1))
```

```
The average error is: 0.11115860000000001
```

Question 2b.

In [39]:

```
def ridgeX(X_data, y_data):
    errors = []

    import math
    for train_index_B, test_index_B in kf8.split(X_data):
        x_trainB = X_data[train_index_B]
        x_holdoutB = X_data[test_index_B]

        y_trainB = y_data[train_index_B]
        y_holdoutB = y_data[test_index_B]

        for train_index, test_index in kf7.split(x_trainB):
            x_train = x_trainB[train_index]
            x_test = x_trainB[test_index]

            y_train = y_trainB[train_index]
            y_test = y_trainB[test_index]

            n,m = X_data.shape
            I = np.identity(m)

            min_w = np.linalg.inv(x_train.T@x_train + 1*I)@x_train.T@y_train
            #print(min_w.shape)
            min_error = error_func(min_w, x_test, y_test)
            #print(min_error.shape)
            for i in range(2,10):
                w = np.linalg.inv(x_train.T@x_train + i*I)@x_train.T@y_train
                #print(w.shape)
                error = error_func(w, x_test, y_test)
                #print(error.shape)
                if((sum(error)/len(error))<(sum(min_error)/len(error))):
                    min_w = w
                    min_error = error


            error_holdout = error_func(min_w, x_holdoutB, y_holdoutB)
            errors.append(error_holdout)
            #print(errors)

    return errors
```

In [64]:

```
avgerror2 = ridgeX(X, y)
print(len(avgerror2))
print((sum(avgerror2)/len(avgerror2))/16)
```

```
56
[-0.00137519  0.00018683  0.0338442  -0.01023701  0.00991959 -0.00719213
  0.00916873 -0.00318349 -0.00531116  0.02146029  0.00094617  0.00334923
  0.00381789  0.01527255  0.0188166   0.01226629]
```

In [65]:

```
list2 = [-0.00137519,  0.00018683,  0.0338442,  -0.01023701,  0.00991959, -0.00719213,
0.00916873, -0.00318349,
         -0.00531116,  0.02146029,  0.00094617,  0.00334923, 0.00381789,  0.01527255,
0.0188166,   0.01226629]
print("The average error is", sum(list2))
```

```
The average error is 0.10174939000000001
```

Question 2c.

In [69]:

```
#generate random linear combination of the original 9 features
new = X@np.random.rand(9, 3)

#append it to the original X feature matrix
X_new = np.hstack((X, new))
```

In [70]:

```
## Repeat Experiment a
avgerror3 = truncatedSVD(X, y)
print(len(avgerror3))
print((sum(avgerror3)/len(avgerror3))/16)
```

```
56
[-0.00583885 -0.02032973  0.02304806 -0.0296349   0.043372   -0.01156517
  0.00634074  0.00404593 -0.00653409  0.06171246 -0.02414963  0.04552239
  0.00570181  0.02030202 -0.00534707  0.00451263]
```

In [71]:

```
list3 = [-0.00583885, -0.02032973,  0.02304806, -0.0296349,   0.043372,   -0.01156517,
0.00634074, 0.00404593,
         -0.00653409,  0.06171246, -0.02414963,  0.04552239, 0.00570181,  0.02030202, -
0.00534707,  0.00451263]

print("The average error is", sum(list3))
```

```
The average error is 0.11115860000000001
```

In [72]:

```
## Repeat Experiment b
avgerror4 = ridgeX(X, y)
print(len(avgerror4))
print((sum(avgerror4)/len(avgerror4))/16)
```

```
56
[-0.00137519  0.00018683  0.0338442  -0.01023701  0.00991959 -0.00719213
  0.00916873 -0.00318349 -0.00531116  0.02146029  0.00094617  0.00334923
  0.00381789  0.01527255  0.0188166   0.01226629]
```

In [73]:

```
list4 = [-0.00137519,  0.00018683,  0.0338442,  -0.01023701,  0.00991959, -0.00719213,
0.00916873, -0.00318349,
         -0.00531116,  0.02146029,  0.00094617,  0.00334923, 0.00381789, 0.01527255, 0.
0188166, 0.01226629]

print("The average error is", sum(list4))
```

```
The average error is 0.10174939000000001
```