In [1]:

```
import numpy as np
import scipy.io as sp
import matplotlib.pyplot as plt

in_data = sp.loadmat("face_emotion_data.mat")
X = in_data['X']
y = in_data['y']
```

1a)

In [2]:

```
w_opt = np.linalg.inv(X.T@X)@X.T@y
print(w_opt)
```

```
[[ 0.94366942]
 [ 0.21373778]
 [ 0.26641775]
 [-0.39221373]
 [-0.00538552]
 [-0.01764687]
 [-0.16632809]
 [-0.0822838 ]
 [-0.16644364]]
```

1b)

If the weight is positive, then the classifier predicts the face to be Happy and if weight is negative then the classifier predicts the face to be Sad. We use the formula, y = sign(x.T@w).

In [3]:

```
#
for i in range(len(w_opt)):
    if( w_opt[i,]> 0):
        print(w_opt[i,], "-> Happy")
    else:
        print(w_opt[i,], "-> Angry")
y_hat = np.sign(X@w_opt)
err = np.mean(y != y_hat)
```

```
[0.94366942] -> Happy
[0.21373778] -> Happy
[0.26641775] -> Happy
[-0.39221373] -> Angry
[-0.00538552] -> Angry
[-0.01764687] -> Angry
[-0.16632809] -> Angry
[-0.0822838] -> Angry
[-0.16644364] -> Angry
```

1c)

The features that have the largest absolute values are the most important. Thus the columns which have the most number of largest absolute value rows are the columns we have to select.

1d)

In [4]:

```
X_new = np.hstack((X[:,0:1],X[:,2:3],X[:,3:4]))
w_opt2 = np.linalg.inv(X_new.T@X_new)@X_new.T@y
y_hat2 = np.sign(X_new@w_opt2)
errOfThree = np.mean(y_hat2 != y)
```

We should choose the features that have the largest absolute value of weight in the corresponding row of the weight matrix. By doing this analysis, we find that, we should choose column 1, 3 and 4.

Procedure for classifier: We need to make a new array with just the required columns 1, 3 and 4. Let's call it X_new. Then we can find a new solution to the least squares problem (w_opt2). Then we can do the same thing which we did in the previous qs and take sign(X_new@w_opt2) to decide the binary classification.

1e)

In [7]:

```
print("Error when using all nine features: ", err*100, "%")
print("Error when using just three of the nine features: ", errOfThree*100, "%")
```

```
Error when using all nine features:  2.34375 %
Error when using just three of the nine features:  6.25 %
```

1f)

In [ ]:

```
for i in range(8):
    Temp = np.hstack((X[i*16,i*16+16:,]))
    NewX = np.hstack((X[0,i*16:,], X[i*16+16,127:,]))

    w_opt3 = np.linalg.inv(NewX.T@NewX)@NewX.T@y
    y_hat3 = np.sign(NewX@w_opt3)

    w_opt4 = np.linalg.inv(Temp.T@Temp)@Temp.T@y
    y_hat4 = np.sign(Temp@w_opt4)

    errNew = np.mean(y_hat3 != y_hat4)
```

In [ ]:

```
```