SML
Assignment2 - Report

1. Computing the global mean and covariance of the data
    - We created two functions to calculate the global mean and the covariance of the data from scratch.

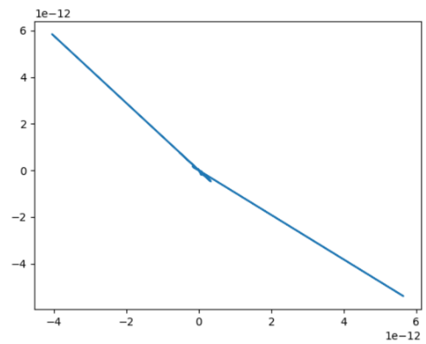
```python
def Mean(data):
    size=len(data);
    dim=len(data[0]);
    X=geek.zeros([dim,1]);

    for j in range(len(data)):
        Y=data[j];
        Y.shape=(dim,1);
        #X+=Y;
        X=np.add(X, Y, out=X, casting="unsafe");

    X=np.divide(X,size);
    print("success");
    return X;

def Covariance(data,Xmean):
    size=len(data);
    dim=len(data[0]);
    C=geek.zeros([dim,dim]);
    for j in range(len(data)):
        Y=data[j];
        Y.shape=(dim,1);
        Temp=Y-Xmean;
        #C+=(Temp.dot(Temp.transpose()));
        C=np.add(C, Temp.dot(Temp.transpose()), out=C, casting="unsafe");
    C=np.divide(C,(size-1));
    print("success");
    return C;
```

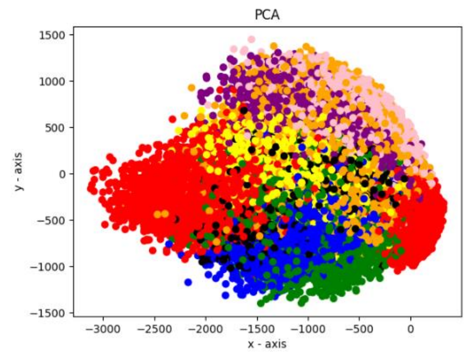
2. PCA and FDA (from scratch)

```python
# find PCA for MNIST dataset

def PCA(data,covariance,energy):

    #find eigenvalue and eigenvector
    eigvalues, eigvectors = la.eig(covariance)


    # find p largest eigenvalue
    idx = eigvalues.argsort()[::-1]
    eigvalues = eigvalues[idx]
    eigvectors = eigvectors[:,idx]
    # reduce to p dimension from 784
    p=find_num_pc(energy,eigvalues)
    eigvectors = eigvectors[:,:p]
    eigv_tran=np.array(eigvectors).transpose()

    Y=eigv_tran.dot(np.array(data).transpose());
    """#retrieve image after PCA
    P=np.linalg.pinv(eigvectors.dot(eigv_tran))
    S=P.dot(eigvectors.dot(Y))
    S_tran=S.transpose()"""

    print("success PCA");
    return Y , eigv_tran;
```

```python
# find FDA for MNIST data
def FDA(data,covariance,cov): #you can add energy

    Sw = cov[0] + cov[1] + cov[2] + cov[3] + cov[4] + cov[5] + cov[6] + cov[7] + cov[8] + cov[9];

    Sb = covariance - Sw;

    Sw_inv = np.linalg.pinv(Sw);

    Ch_eq = Sw_inv.dot(Sb);

    #find eigenvalue and eigenvector of characteristic equation

    eigvalues , eigvectors = la.eig(Ch_eq);

    # find p largest eigenvalue
    idx = eigvalues.argsort()[::-1]
    eigvalues = eigvalues[idx]
    eigvectors = eigvectors[:,idx]

    # reduce to p dimension from 784
    #p=find_num_pc(energy,eigvalues)
    eigvectors = eigvectors[:,:9] # there are 10 classes

    eigv_tran=np.array(eigvectors).transpose()

    Y=eigv_tran.dot(np.array(data).transpose());
    print("succes FDA");
    return Y , eigv_tran;
```

3. Visualizing data using a scatter plot (after applying PCA & FDA)





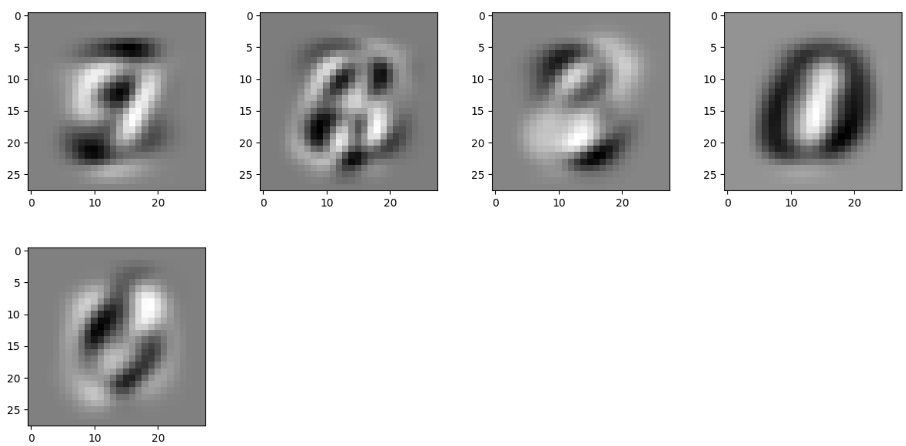
4. LDA discriminant function (from scratch)

```
def LDA(S,mean,cov,P):


    g=geek.zeros([10,1])
    g[0]=-(1/2)*(((S-mean[0]).transpose()).dot((np.linalg.pinv(cov[0])).dot(S-mean[0]))) + np.log(P[0]);
    g[1]=-(1/2)*(((S-mean[1]).transpose()).dot((np.linalg.pinv(cov[1])).dot(S-mean[1]))) + np.log(P[1]);
    g[2]=-(1/2)*(((S-mean[2]).transpose()).dot((np.linalg.pinv(cov[2])).dot(S-mean[2]))) + np.log(P[2]);
    g[3]=-(1/2)*(((S-mean[3]).transpose()).dot((np.linalg.pinv(cov[3])).dot(S-mean[3]))) + np.log(P[3]);
    g[4]=-(1/2)*(((S-mean[4]).transpose()).dot((np.linalg.pinv(cov[4])).dot(S-mean[4]))) + np.log(P[4]);
    g[5]=-(1/2)*(((S-mean[5]).transpose()).dot((np.linalg.pinv(cov[5])).dot(S-mean[5]))) + np.log(P[5]);
    g[6]=-(1/2)*(((S-mean[6]).transpose()).dot((np.linalg.pinv(cov[6])).dot(S-mean[6]))) + np.log(P[6]);
    g[7]=-(1/2)*(((S-mean[7]).transpose()).dot((np.linalg.pinv(cov[7])).dot(S-mean[7]))) + np.log(P[7]);
    g[8]=-(1/2)*(((S-mean[8]).transpose()).dot((np.linalg.pinv(cov[8])).dot(S-mean[8]))) + np.log(P[8]);
    g[9]=-(1/2)*(((S-mean[9]).transpose()).dot((np.linalg.pinv(cov[9])).dot(S-mean[9]))) + np.log(P[9]);

    min=0;
    for i in range(1,10):
        if(g[i]>g[min]):
            min=i;
        else:
            continue;
    return min;
```

5. PCA with 95% eigen energy on MNIST and LDA for classification and the accuracy on test data.

    Accuracy: 89.3 % on 1000 test data.

6. Visualizing and analyzing eigenvectors obtained using PCA.



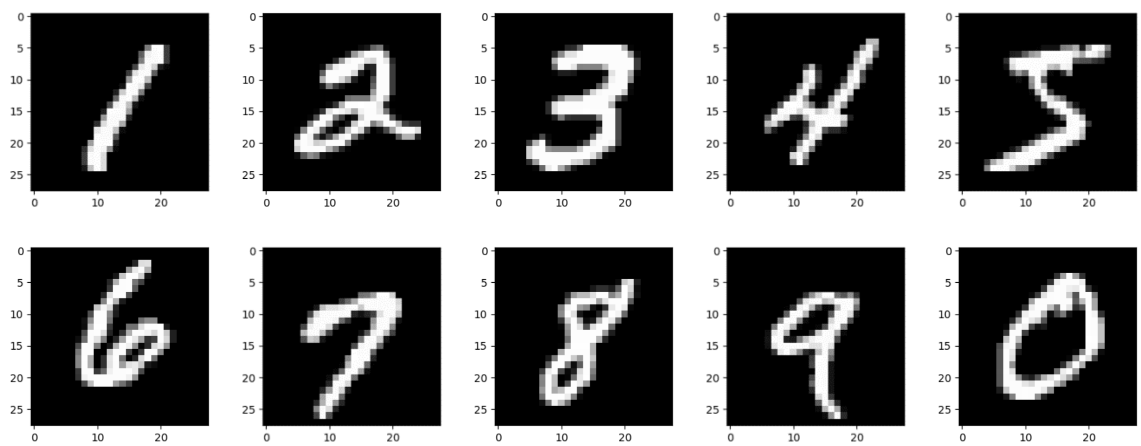
7. Applying PCA with Different eigen values and comparing and analyzing on accuracy.
    a) 70% eigen energy
    - 92.6% on 1000 test data
    b) 90% eigen energy
    -91.4% on 1000 test data
    c) 99% eigen energy
    -86.9% on 1000 test data

8. FDA on MNIST and then LDA for classification and reporting the accuracy on test data.
    Accuracy: 24% on 100 test data

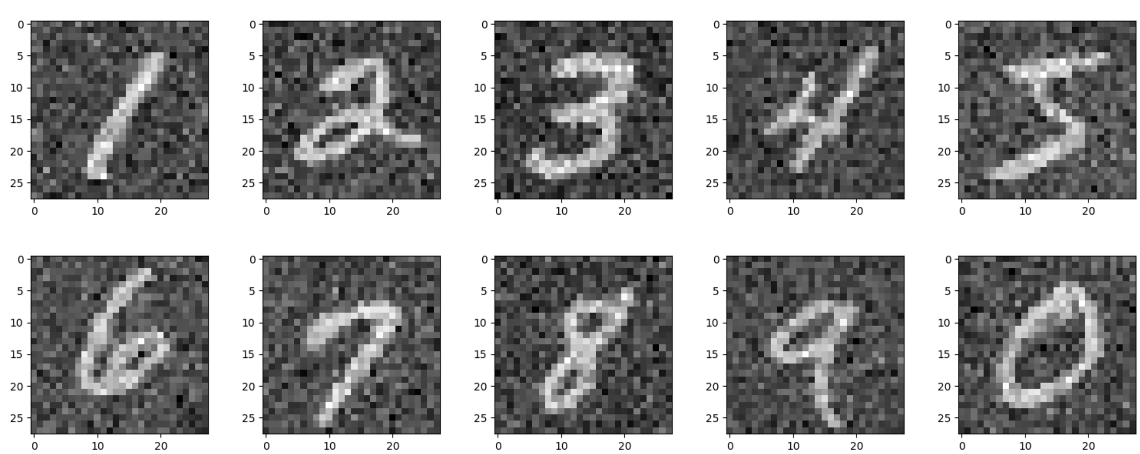
9. Perform PCA then FDA. Classifying the transformed datasets using LDA. Analyzing the results on accuracy.

Accuracy: 87% on 1000 test data.

Original Trained Data



Data Corrupted with Gaussian Noise *(mu = 0, sigma = 50)* on Trained Data



Result After Linear PCA