SMAI Q1 REPORT
Himansh Sheoran
20161211

Problem Statement

Given a set of images we have to perform Principal component analysis on it and reduce it dimension and also find the MSE between original images an reconstructed images.

Preprocessing

Before moving on, we need to reduce the size of the dataset as the machine cannot e the given dataset.
Reduce all images to the size of 32*32 so as to make the model processable.
Total number of features now in the model is 1024

PCA_INFORMATION

Principal component analysis (PCA) is a technique used for identification of a smaller number of uncorrelated variables known as principal components from a larger set of data. So in PCA if we want to capture the data in low dimension say from D to K dimension(k<D) We will first find the K dimension plane(in 1-d its a line) .Plane or line will be such that the variance of the dataset from the plane or line is maximum.And then we will project our data onto that plane and thus reducing the dimensionality from N to K

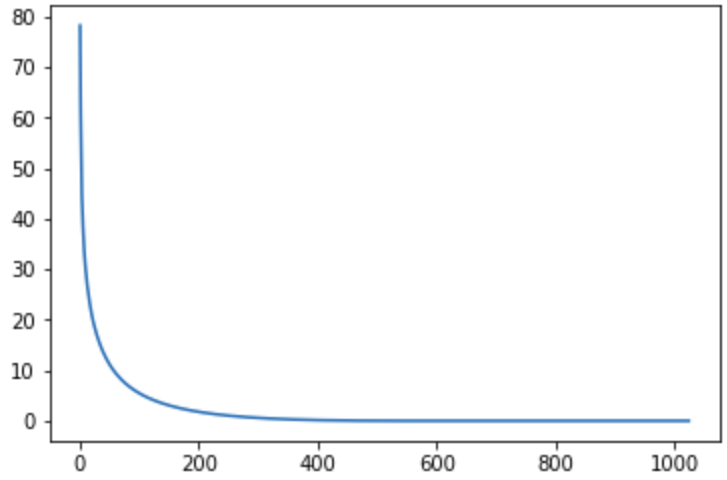
Implementation of PCA

As I have 1024 features of each image to deal with, I proceed to deal with PCA.
First normalize the data by subtracting the mean of all the features from it.
Second applied the single value decomposition to the image model to get a representation of all the images in a 2D matrix of size N*D where d is the number of total dimensions.
Out of the output obtained from the matrix after single matrix decomposition i took the eigenvectors which are already in sorted manner as it is the property of the single value decomposition.
Now since each point can be written in terms of eigenvector so i used this eigenvectors to find those coefficients by which i can the write the data points as a1*lambd1 + a2*lambda2 ..... a1,a2... are coefficients and lambda1, lambda2... are eigenvectors
Formula used =
coefficients = np.matmul(mean1 , eigenvectors.T)

Then to project the data in k dimension i have taken first k coefficients.
Now to reconstruct it back i used this
Formula
arr = np.matmul(coefficients[:,:k],eigenvectors[:k,:])

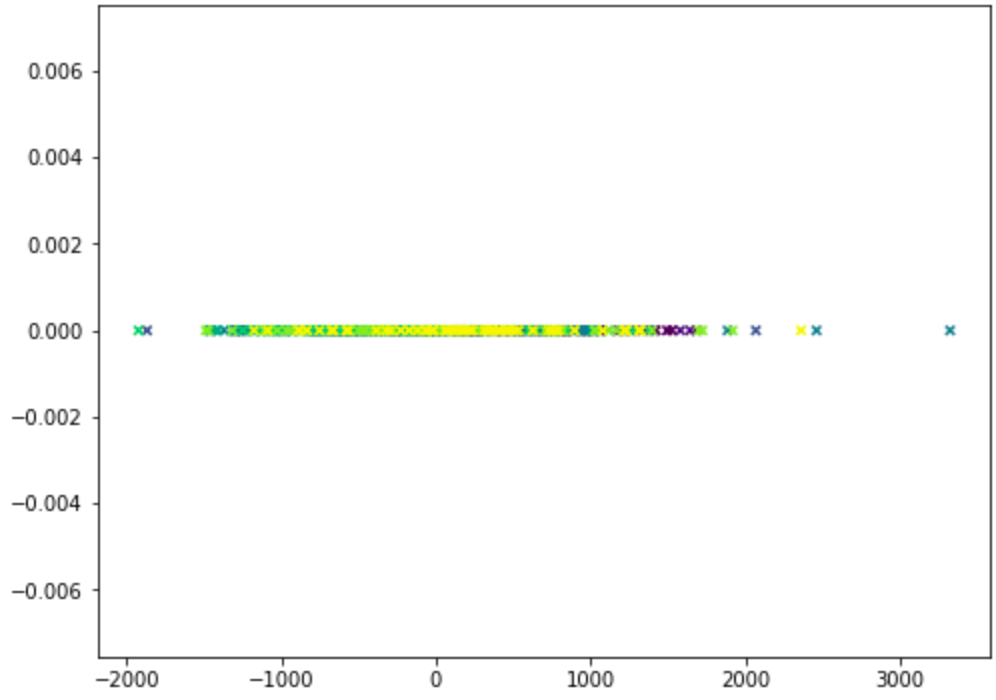
this gives the array of NxD back , which is nothing just multiplying it back with eigenvector.Transpose.

Plot of Mean Square Error for each K

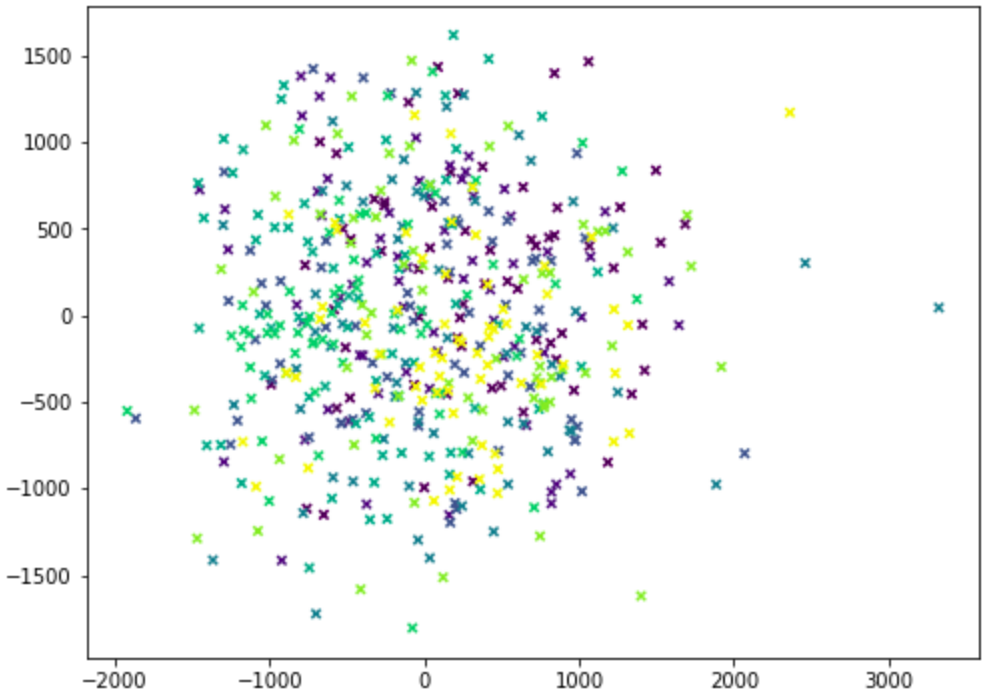


Scatter Plots for 1D, 2D and 3D representation of data

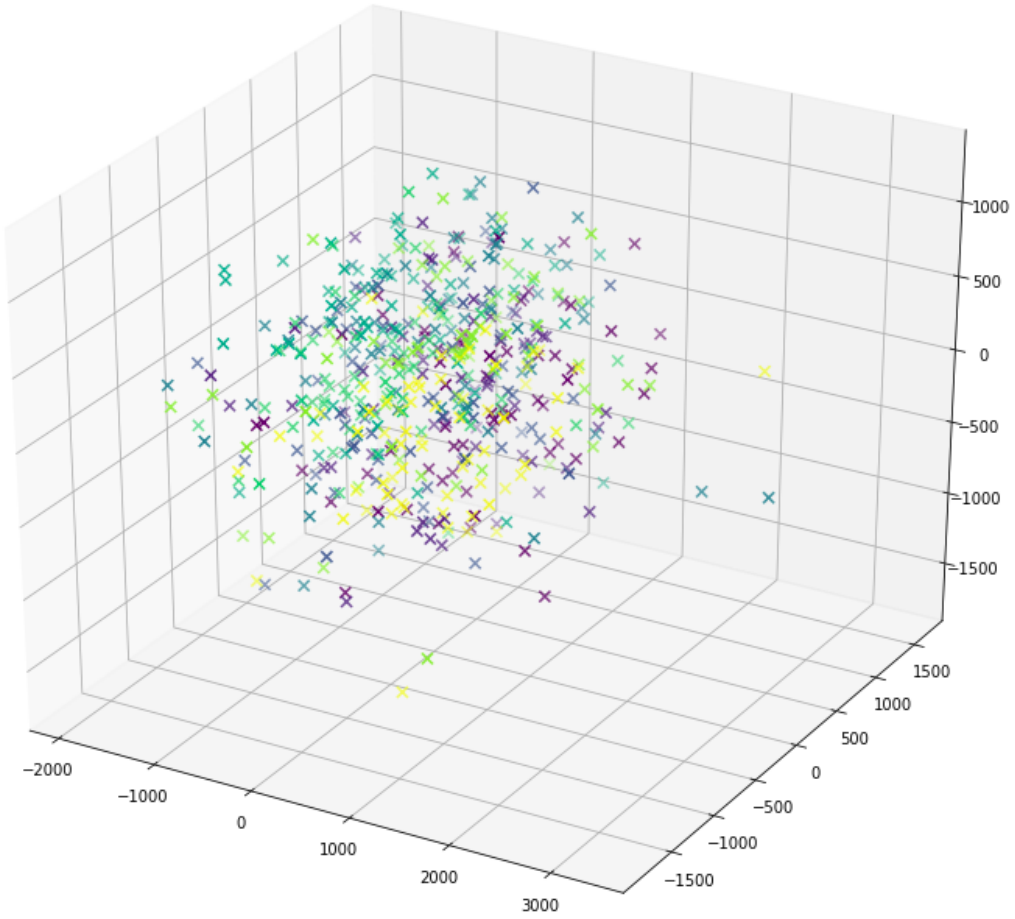
1D representation



2D representation



3D representation


1000
500
0
−500
−1000
−1500
1500
1000
500
0
−500
−1000
−1500
−2000
−1000
0
1000
2000
3000