1. **Digit Clustering**

a. I've selected KMeans clustering method for the digit data set. This algorithm accepts two inputs. The data itself, and "k", the number of clusters. The output is k clusters with input data partitioned among them. The aim of K-means (or clustering) is this : We want to group the items into k clusters such that all items in same cluster are as similar to each other as possible. And items not in same cluster are as different as possible. We use the distance measures to calculate similarity and dissimilarity. One of the important concept in K-means is that of centroid. Each cluster has a centroid. You can consider it as the point that is most representative of the cluster. Equivalently, centroid is point that is the "center" of a cluster. An algorithm for partitioning (or clustering) N data points into K disjoint subsets S_j containing N_j data points so as to minimize the sum-of-squares criterion

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} |x_n - \mu_j|^2,$$

where x_n is a vector representing the nth data point and μ_j is the geometric centroid of the data points in S_j.

Advantages :

1. The strength of Kmeans algorithm lies in its computational efficiency and the nature of easy to use. If variables are huge, then K-Means most of the times computationally faster than hierarchical clustering, if we keep k smalls.

2) K-Means produce tighter clusters than hierarchical clustering, especially if the clusters are globular.

Disadvantages :

1) Difficult to predict K-Value.

2) With global cluster, it didn't work well.

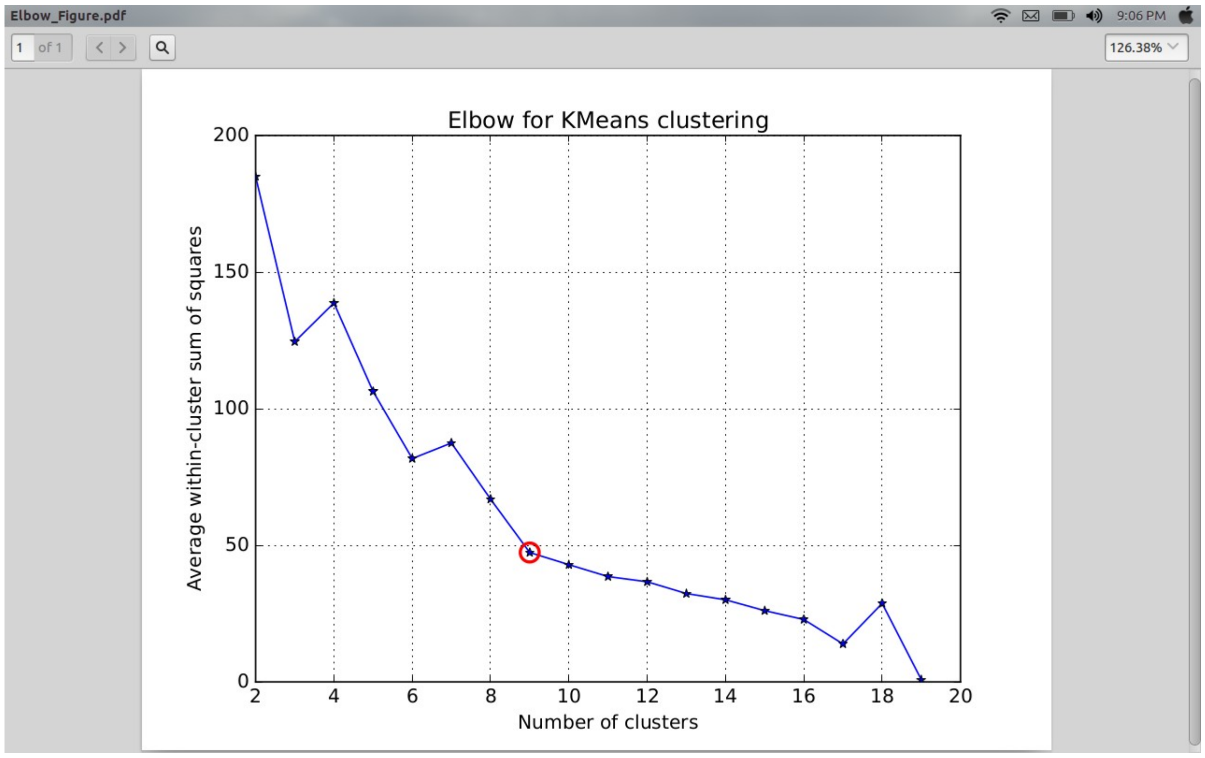
3) Different initial partitions can result in different final clusters.

4) It does not work well with clusters (in the original data) of Different size and Different density.

b. I'm using dispersion i.e. sum-of-squares of difference between cluster centroid and data points belongs to that particular cluster. It basically defines how much a cluster holds it data points tight. It is measured by finding the euclidean distance between each data points and cluster centroid. Main motto behind using this method was that it is very fast to execute whereas other method such as gap statistics are very slow and performance was not that good.

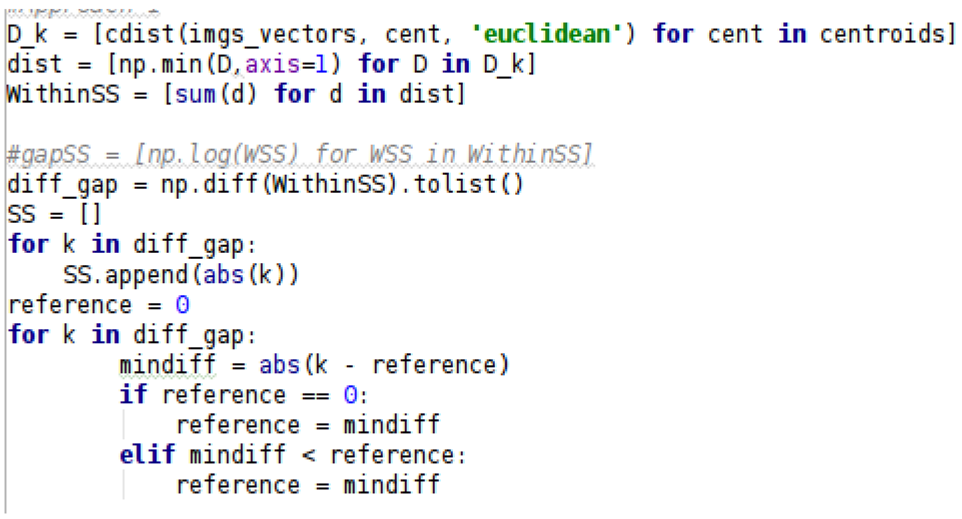
$$SSE = \sum_{i=1}^{K} \sum_{x \in c_i} dist(x, c_i)^2$$

c)



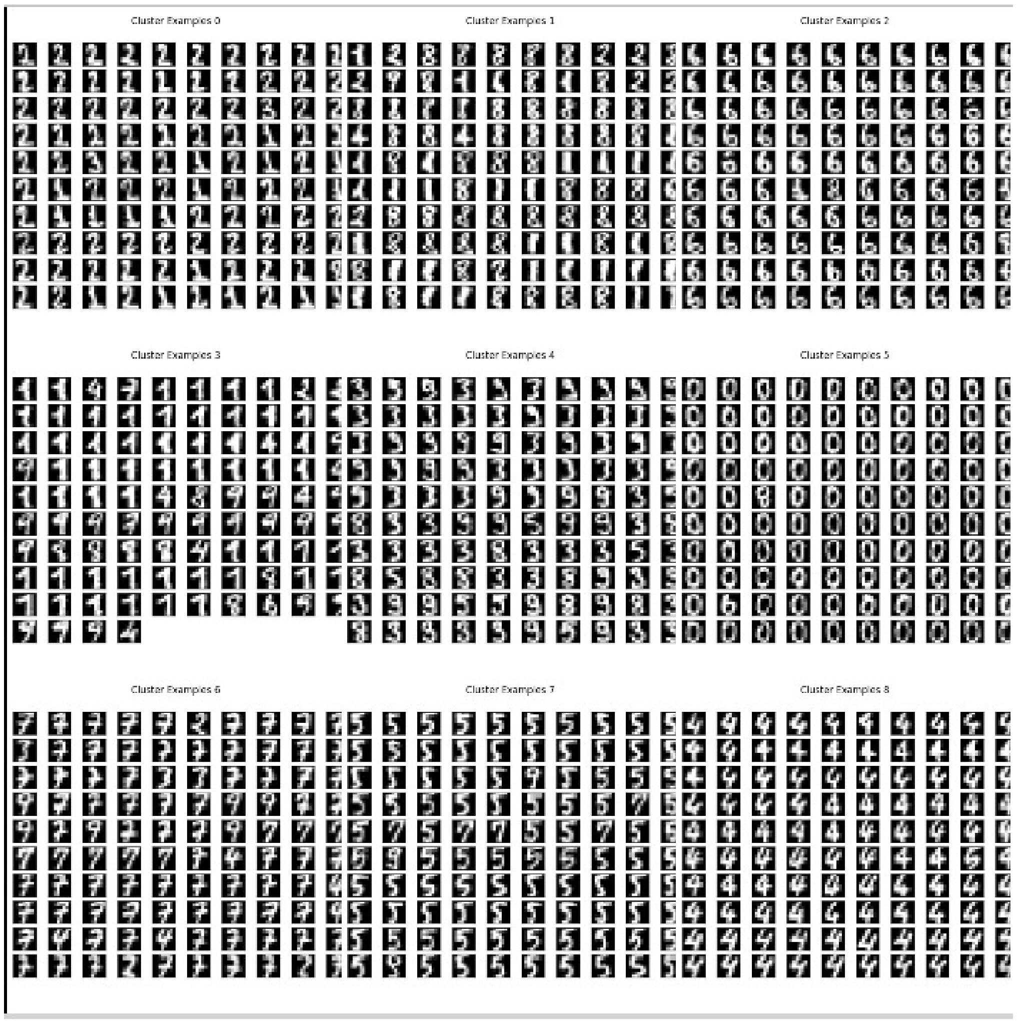
d. *In order to find optimal number of cluster, I'm observing the second derivative of sum of squared error. Main intuition behind using this method was to observe the minimum change in error for every increase in the number of cluster.*

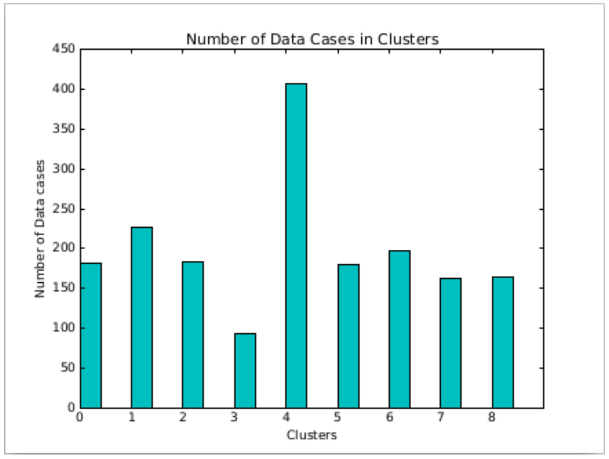
e. Optimal number of cluster found = 9 which is less than number of digits in the sample (i.e. 0-9)

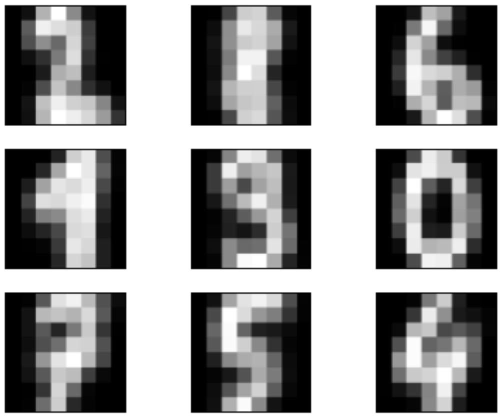


```
D_k = [cdist(imgs_vectors, cent, 'euclidean') for cent in centroids]
dist = [np.min(D,axis=1) for D in D_k]
WithinSS = [sum(d) for d in dist]

#gapSS = [np.log(WSS) for WSS in WithinSS]
diff_gap = np.diff(WithinSS).tolist()
SS = []
for k in diff_gap:
    SS.append(abs(k))
reference = 0
for k in diff_gap:
        mindiff = abs(k - reference)
        if reference == 0:
            reference = mindiff
        elif mindiff < reference:
            reference = mindiff
```







f) As we know optimal number of cluster found is 9. It is less than the corresponding 0-9 digits. It may be because as we can see above Kmeans algorithm has distributed 9 into clusters of 1, 3, 7 and 8. Kmeans calculates centroid as mean of the datapoints 9 never got the majority.

2. Dimensionality Reduction for Image Denoising:

a. I've selected PCA dimensionality reduction method for image denoising. PCA finds the principal components of data, which gives the directions where there is the most variance, the directions where the data is most spread out. The sample estimate of the variance in the direction w given the data set X is given by the expression:

$$\frac{1}{N}\sum_{i=1}^{N}(\mathbf{X}_i\mathbf{w}-\mu)^2 \quad \text{where} \quad \mu = \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_i\mathbf{w} \qquad \mathbf{w} = \sum_{d=1}^{D} \omega_d \mathbf{V}_d = \mathbf{V}_1$$

Here **w** gives the direction of maximum variation hence the principal component.

Advantages:

Its low noise sensitivity, the decreased requirements for capacity and memory, and increased efficiency given the processes taking place in a smaller dimensions.

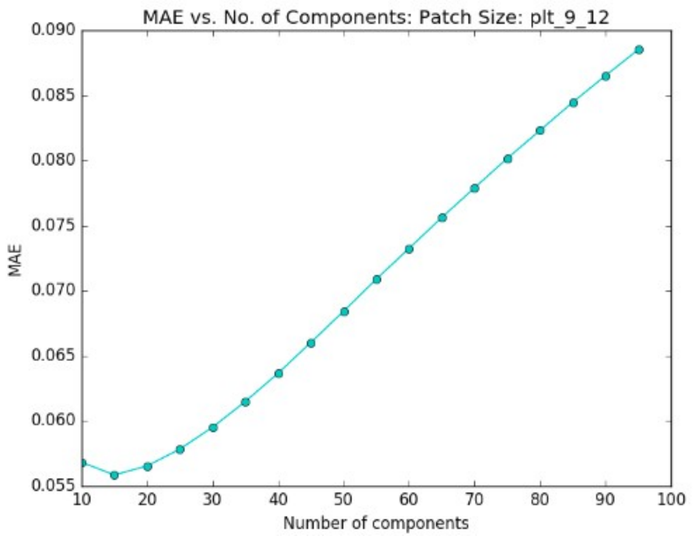
Disadvantages:

1) Lack of redundancy of data given the orthogonal components.

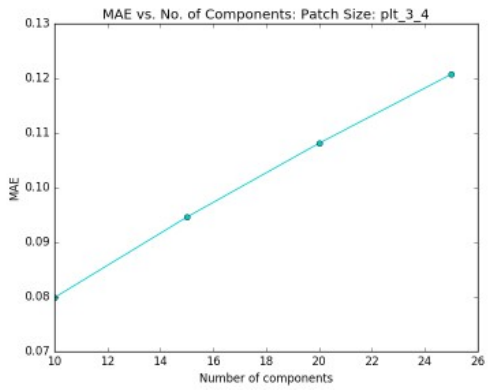
2) Reduced complexity in images' grouping with the use of PCA.

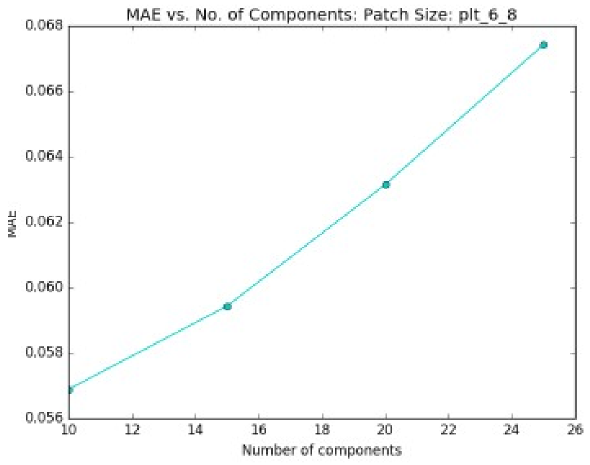
3) Reduction of noise since the maximum variation basis is chosen and so the small variations in the background are ignored automatically.

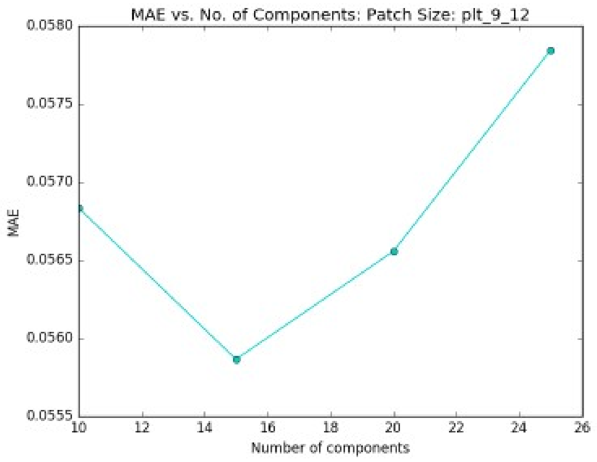
b) I've selected patch sizes of [(3,4), (6,8), (9,12), (12,16), (15,20)] and number of components as the primary hyper-parameters of the PCA dimensionality reduction method. In order to preserve the same expect ratio I'm keeping the ratio of 3:4 for every combinations as the image has a resolution of 900*1200. Between components range of 10 and 30 we can see minimum MAE.

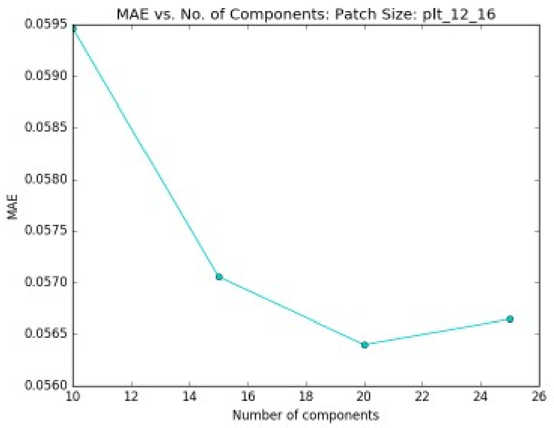


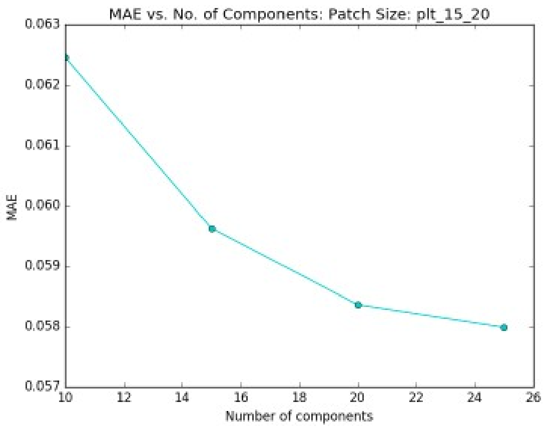
c)

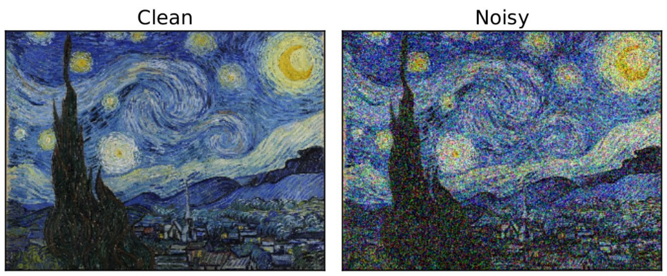

MAE vs. No. of Components: Patch Size: plt_3_4
MAE
Number of components

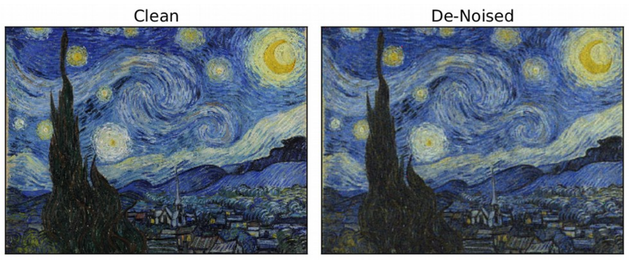

MAE vs. No. of Components: Patch Size: plt_6_8
MAE
Number of components


MAE vs. No. of Components: Patch Size: plt_9_12
MAE
Number of components


MAE vs. No. of Components: Patch Size: plt_12_16
MAE
Number of components


MAE vs. No. of Components: Patch Size: plt_15_20
MAE
Number of components


Clean
Noisy


Clean
De-Noised