Unsupervised Learning Report Ashwini Iyer, CS 4641, Project 3, March 23, 2018

Dataset Descriptions

The two datasets used in this experiment are the Iris and MNIST datasets. I also used MNIST in both project 1 and project 2. The Iris data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. The Iris setosa, Iris virginica and Iris versicolor. One class is linearly separable from the other 2; the latter are not linearly separable from each other. This and its low dimensionality make it well suited for classification problems. The full MNIST dataset was used for visualization and contains images of handwritten digits with approximately 50,000 training images and 10,000 testing images, where each sample is a 28 x 28 image with no color channels. I also used sklearn's reduced MNIST dataset which was a 8 x 8 image with no color channels and 1,797 training examples for neural network training and clustering performance evaluations. MNIST is quite suited to dimensionality reduction as it is highly-dimensional and is hard to visualize effectively along any dimensions. Clustering algorithms on MNIST without dimensionality reduction is expected to perform poorly due to its high dimensionality. I decided to use it because, due to its high dimensionality, I was curious to see if dimensionality reduction would solve this problem.

Clustering Algorithms

I primarily used the sklearn library for the clustering algorithms and dimensionality reduction algorithms and ABAGAIL to run the neural networks. For the purpose of analysis and experimental consistency both Expectation Maximization and K-means used Euclidean Distance to measure distances between instances. I could have used Manhattan distance which works better for very high dimensional vectors, but the Iris dataset only has 4 features and Euclidean Distance is more standard for image datasets.

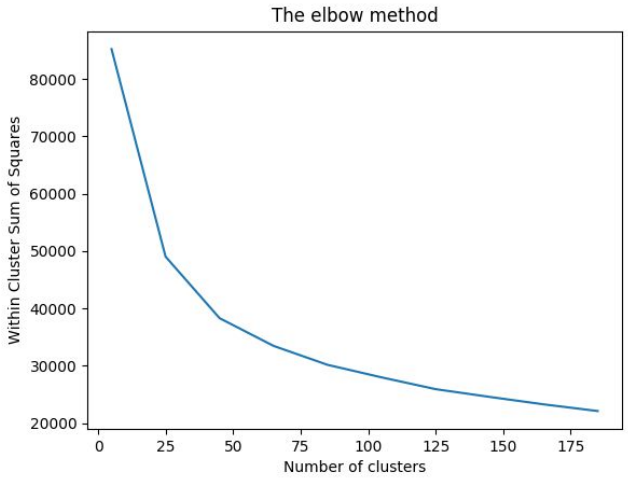
K-means

K-means is a naive unsupervised learning algorithm that groups a dataset into 'k' clusters. It is naive because it will cluster the data regardless of whether the inputted 'k' is the right number of clusters for the dataset. In order to validate whether the selected value of 'k' is suitable for the dataset, I used the elbow method. By running K-means clustering on each dataset for a range of clusters and calculating the sum of squares errors for each, I was able to determine the suitable 'k' cluster values by looking at the graph and identifying where k has a small value but also a low sum of squares error, this is called the 'elbow'.

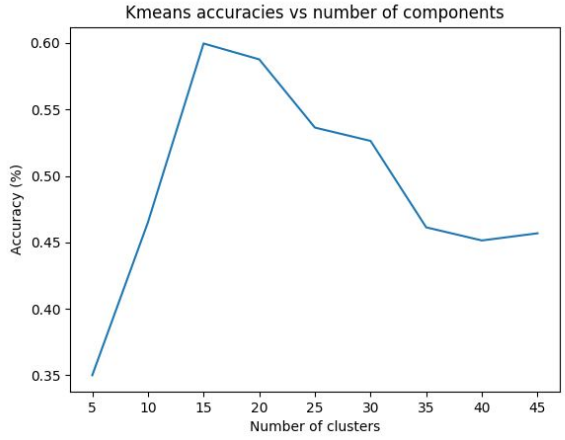
<u>MNIST</u>

This graph shows the relationship between the sum of squares and number of clusters for the MNIST dataset. The graph shows that a faint elbow begins around 25. Due to the fact that the elbow was not very clear, I decided to further analyze with a smaller range and calculated the accuracy of clusters ranging from 5 to 45, a noticeably smaller range than the previous 0 to 175

clusters. From the graph and table below it is evident that the peak in accuracy exists at 15, not



25 as previously concluded. Now that we have determined the ideal k for the MNIST dataset, it is important to consider that even with a k=15 the Accuracy % is only 59.9% which is very poor. Also 15 is a much larger number of clusters than the associated labels for the number of unique digits in the dataset, which is 10. This is because the raw form of the data is 784 pixels, which is much harder to cluster without dimensionality reduction than the 4-dimensional iris dataset. This is because of the dimensionality of the MNIST dataset, we would need to perform dimensionality reduction to improve this. Because by lowering the dimensionality, the clustering algorithms will be able to better identify clusters and it would reduce variance.

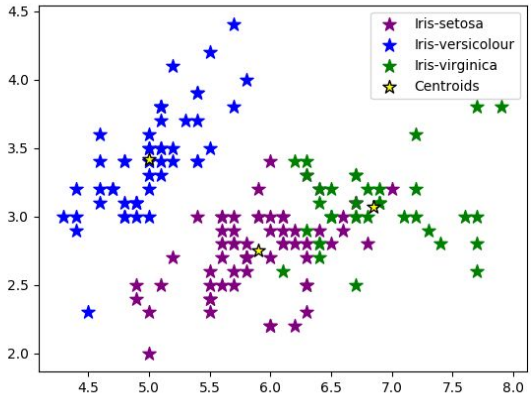


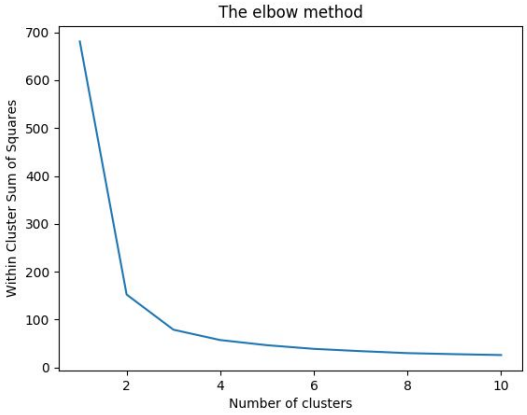
Clusters	Accuracy %
5	35%
10	46.5%
15	59.9%
20	58.7%
25	53.6%
30	52.6%
35	46.1%
40	45.1%
45	45.6

For MNIST, we cannot visualize the clusters we get because they have too high dimensionality but you can visualize them after dimensionality reduction.

IRIS

To figure out the ideal ‘k’ value to use for the Iris Data set, I did the elbow method again. This time it was clear that the elbow existed between 2 and 3. This makes sense because the Iris dataset has 3 classes, 2 of which are linearly separable.

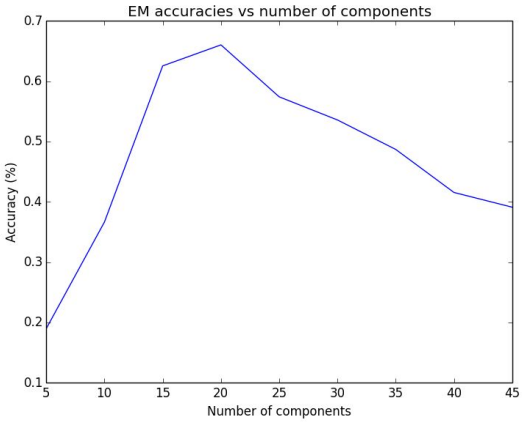




The graph on the left is produced by plotting the data points in the IRIS dataset along the first two dimensions after performing K-means clustering. This allows us to visualize the clusters. The three centroids calculated make sense as they are approximately the center of the various classes of flowers. Without labeling, K-means clustering was able to approximate the true labels of the classes and group the various flower classes properly. The same visualization is not performed for the high dimensional MNIST and visualizing along 2 dimensions out of the high dimensional data is not feasible, and instead the data is visualized later in the dimensionality reduction section.

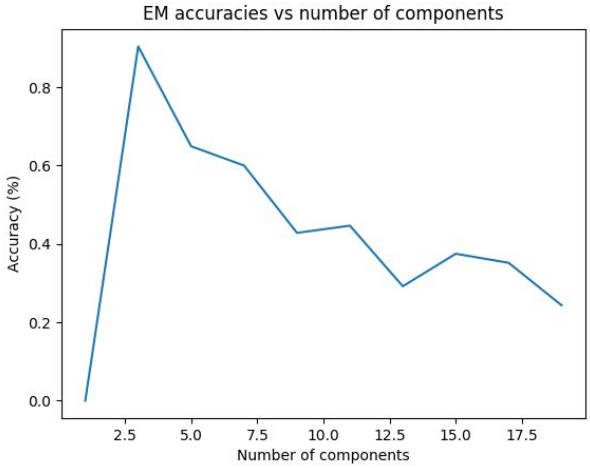
Expectation Maximization

MNIST



Clusters	Accuracy %
5	18.9%
10	36.6%
15	62.5%
20	66.01%
25	57.40%
30	53.57%
35	48.68%
40	41.5%
45	39.1

IRIS



Clusters	Accuracy %
1	0%
3	90.3%
5	64.9%
7	59.9%
9	42.8%
11	44.6%
13	29.18%
15	37.47%
17	35.16%
19	24.32%

Expectation Maximization and K-means Clustering agreed on the natural cluster number for the MNIST and IRIS datasets. The optimal cluster number was around 15-20 components for MNIST and strictly 3 for IRIS. The IRIS dataset only has 4 features and 3 classes, 2 of which aren't linearly separable. The cluster accuracy decreases when more clusters than the number of features that exist are attempted to be formed, and 3 seems to have the highest accuracy of 90.3% and 62.0% as the natural clusters mimic the 3 classes as visualized below using dimensionality reduction to reduce the number of features and visualize them in 3D or 2D.

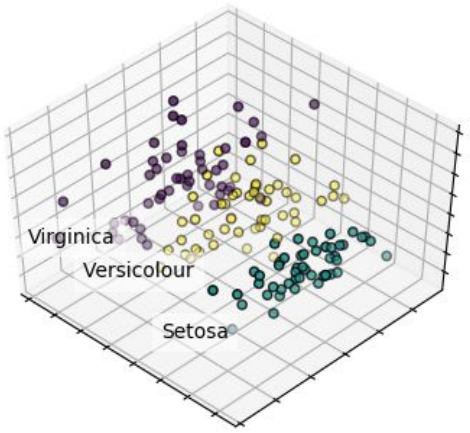
I iterated over the hyperparameters in the clustering algorithms which shared in common the number of components and used the accuracy graph and the elbow method to determine the optimal number of components. Additionally, I also varied the hyperparameters of centroid initialization for K-means, number of iterations, random states, and covariance type for the EM algorithm.

Dimensionality Reduction Algorithms

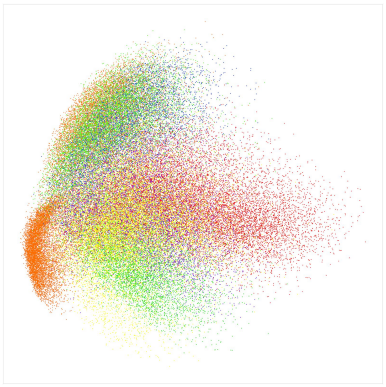
For dimensionality reduction in MNIST, for the sake of visualization, I used the full MNIST dataset reducing 784 dimensions down to 2 dimensions. When doing clustering after dimensionality reduction, I used the sklearn's digits dataset and reduced to 20 dimensions, which is the value found using the elbow method. For Iris, I reduced 4 dimensions down to 3 for both visualization and for data pre processing before clustering.

PCA

IRIS



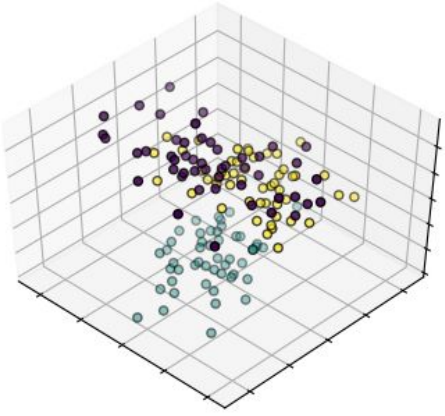
MNIST



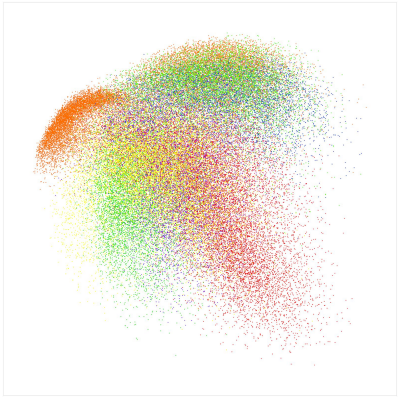
For Iris after PCA the data looks clean and the 3 classes are easily distinguishable, it is also clear that Virginica and Versicolour are the two linearly inseparable classes. The same goes for MNIST after PCA, we can clearly identify or differentiate most of the 10 digit categories based on color. The Eigenvalues for PCA on Iris were: 4.2248, 0.2422, 0.0785 and the eigenvalues on MNIST were: 0.002753 and 0.002518. The amount of eigenvectors/values that exist equals the number of dimensions the data set has. So for the Iris data set there are 3 which makes sense because it was reduced to 3 dimensions, same goes for MNIST because it went down to 2 dimensions and has two eigenvalues. The eigenvectors of PCA give us a more useful axis to frame the data in. The eigenvalues for Iris were heavily skewed, so only one axis shows actual variance the rest are redundant or less relevant. This makes sense because along the Iris dataset looking at sepal length or petal length, one axis is enough to explain the variance. On the other hand for MNIST as both eigenvalues are close to zero but with similar magnitude, so they explain variation in the new dimension equally.

ICA

IRIS



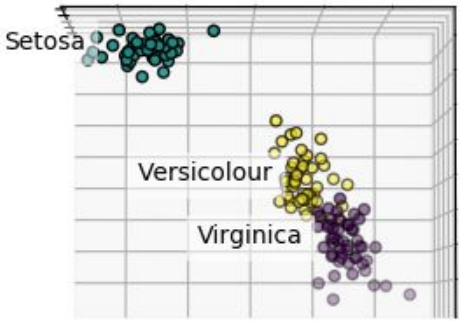
MNIST



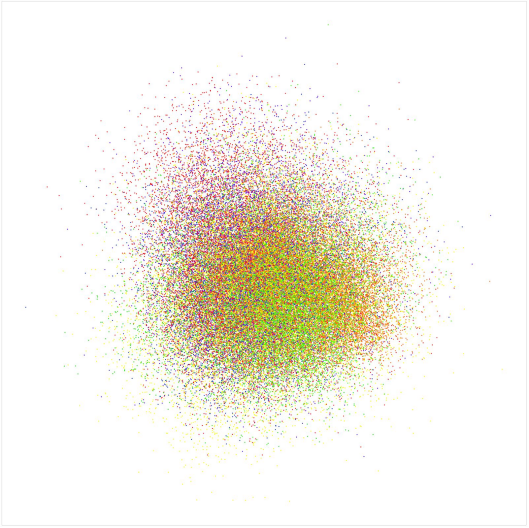
For ICA, I used the FastICA python scikit learn library. Looking at the Iris graph we can identify the different classes by their colors relatively, but not nearly to the same extent that we could when we ran PCA. This is because here we want to find a reduced rank solution which is PCA's focus and not a representation of our data that is independent of sub-elements which is ICA's focus. For MNIST, ICA worked very well. We are able to pretty clearly see the different color groups that represent the different digit groups. This means that after generating projections the data was not reconstructed well.

Randomized Projections

IRIS



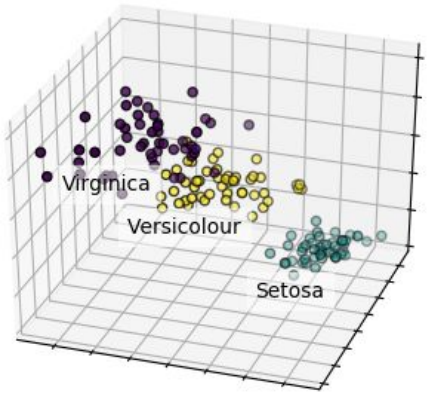
MNIST



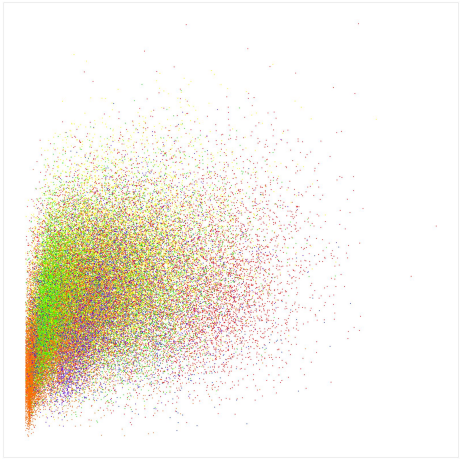
For Iris, the setosa data points are very clearly separated from the other two classes of data. This makes sense as setosa is linearly separable from the other two classes whereas versicolour and virginica are not. Running randomized projection multiple times created some variation but the setosa class was still highly separated from the other two. Unfortunately, this did not perform as well on MNIST, the digits are not distinguishable from each other by color. This means that after generating projections the data was not reconstructed well. Random Projection has preserved a similar skew and kurtosis with the new distribution on two dimensions, but it hasn't preserved similar means and variances. Random Projection suffered when the number of components was very small for the sake of visualization but performed well with 15 components (after a hyperparameter search) and resulted in high accuracies for the clustering algorithms in Table 1.

Feature Agglomeration

IRIS



MNIST



For Feature Agglomeration on Iris it seems that we can clearly identify Setosa as linearly separable to the other two classes. The fact that we can do this also allows us to conclude that after generating projections the data was reconstructed well. For MNIST, we cannot clearly identify the color groups of the digits, which means that the data was not reconstructed well after generating projections. That being said it is still more readable than the MNIST result for Feature Agglomeration above allowing us to conclude that this is a better approach that Feature Agglomeration for MNIST style datasets.

Clustering After Dimensionality Reduction Results & Further Analysis

I performed the clustering algorithms and dimensionality reductions and calculated an unpaired label accuracy metric using the Rand Index which computes a similarity measure between two clusterings as the cluster labels may not directly correspond to the data labels. The most common label in a cluster would give the associated cluster that label. When I reproduced the clustering experiments on the datasets projected onto the new spaces created by ICA, PCA and RP, I verified that I did get similar clusters as before. This is because they resemble the number of classes in the dataset.

Table 1. Clustering and Dimensionality Reduction on Accuracy

Dataset	Dimensionality Red.	Clustering Alg.	#Clusters	% Accuracy
MNIST	None	K-means	15	59.770%
MNIST	None	EM	15	62.53%
MNIST	PCA	K-means	15	55.40%
MNIST	RP	K-means	15	40.41%

MNIST	ICA	K-means	15	57.90%
MNIST	FA	K-means	15	48.31%
MNIST	PCA	EM	15	**63.74%**
MNIST	RP	EM	15	41.315%
MNIST	FA	EM	15	49.44%
MNIST	ICA	EM	15	62.06%
Iris	PCA	K-means	3	62.01%
Iris	RP	K-means	3	92.21%
Iris	FA	K-means	3	57.10%
Iris	ICA	K-means	3	57.13%
Iris	PCA	EM	3	88.59%
Iris	RP	EM	3	**94.10%**
Iris	FA	EM	3	90.3%
Iris	ICA	EM	3	88.59%
Iris	None	EM	3	90.38%
Iris	None	K-means	3	62.01%

Comprehensive Comparison of Techniques

As shown by the first two rows of the table, Expected Maximization (EM) results in a higher accuracy than K-means for both data sets, resulting in 59% vs 62% for MNIST and 90% vs 62% for Iris. This is because K-means hard assigns a data point to one particular cluster on convergence and makes use of the L2 norm when optimizing centroid coordinates. On the other hard EM soft assigns a point to clusters meaning it gives a probability of any point belonging to any centroid and is based on Expectation which is the probability of the point belonging to a particular cluster. As a result, K-means is more biased towards spherical clusters, whereas our MNIST data does not resemble that particular cluster distribution. Similarly, the Iris dataset is not composed of spherical clusters, and EM has better performance than K-means. Also the clustering algorithms without Dimensionality Reduction perform poorly on the MNIST dataset due to MNIST's high dimensional nature, whereas they perform decently on the IRIS dataset because the full dataset already has few dimensions. Dimensionality reduction improves MNIST

performance greatly because of this dimensionality reduction. Clustering algorithms perform better on low-dimensional data where the dimensions explain more of the variance in the original data.

Now, I will compare the performances of the clustering algorithms with the dimensionality reduction algorithms on each dataset. For MNIST, EM achieved a 62.53% accuracy with no dimensionality reduction. Whereas it scored the following with the algorithms: 63.74% with PCA, 41.31% with RP, 62.06% with ICA, 49.44% with FA. There is an increase in performance when using PCA to compress that data rather than separate the data like ICA. Additionally, there were no increases in performance when dimensionality reduction was performed before K-means on Mnist, which indicates that the dimensionality reduction of 64 dimensions down to 15 dimensions achieved the same performance as clustering directly from 64 dimensions for K-means. Now, let's consider Iris. K-means on Iris with no dimensionality reduction achieved a 62.01% accuracy. But its accuracy improved with Randomized Projection, achieving a 92% accuracy. Similarly, RP on EM performed the best on the Iris data with 94.10%. Randomized Projection may be particularly suitable for the Iris data where points in the higher dimensions are projected to three dimensions while preserving the distances between the points. Clustering in the full dimensions of the Iris data based on distances is actually quite suitable as the features are quite different among the various classes. As a result, it makes sense than an approach that seeks to preserve this distance, performs well.

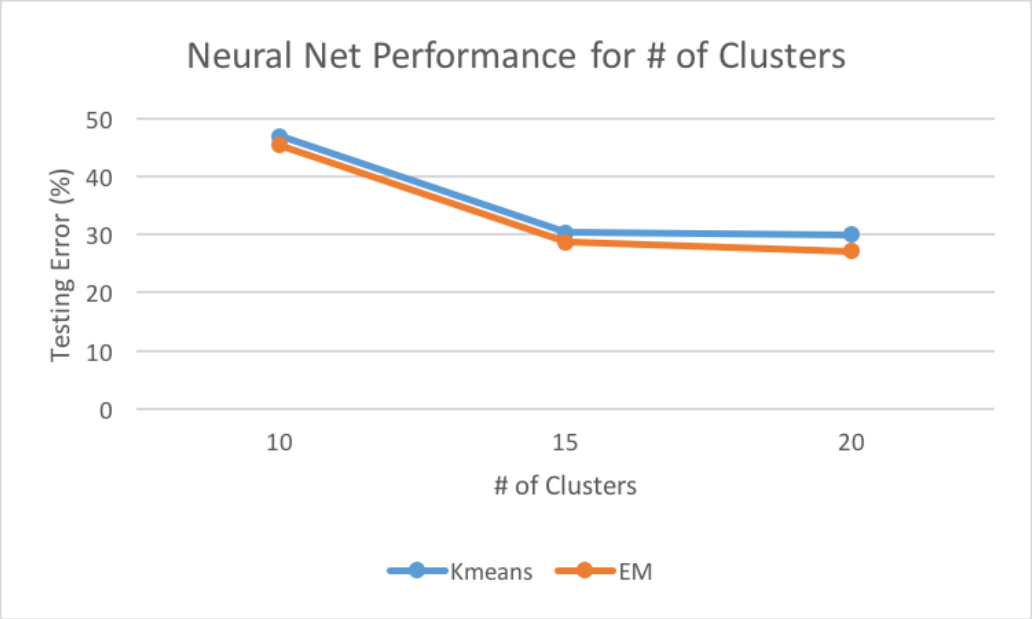
Neural Network

Table of Neural Net testing accuracy with clustering and dimensionality reduction pre-processing on MNIST data.

	# of Features	Time Elapsed (s)	Test error (%)
Original Data	64	1.025	6.3
PCA	15	1.070	**5.9**
ICA	15	1.908	6.3
Feature Agglomeration	15	2.371	8.9
Random Projection	15	2.496	18.7
K-means (20 clusters)	1	1.992	30
EM (20 clusters)	1	1.996	27.2

PCA which does data compression seems to have helped neural network performance by reducing the test error to 5.9%. The other dimensionality reduction methods have sacrificed

some performance and while each iteration runs faster as there is less data, the overall neural network takes more steps to converge and more total time elapses.



The neural network testing error decreases with the number of clusters for both K-means and EM from 45% to around 30%. For MNIST, there are 10 unique classes of data, but unsupervised learning algorithms may not be able to cluster into these 10 categories perfectly. For example, sometimes a groups of 1's and 7's may look similar to each other and may form their own cluster. As the number of clusters decreases for feature input into the neural network, these groups collapse into the same feature labels, resulting in loss of detail and resulting accuracy. The clustering algorithms relabeled the feature points with the corresponding clusters that each data point belonged to, resulting in a single feature. It is quite amazing to see how 64 features or even 784 features can be collapsed into a single feature and still achieve a 73% accuracy in a digit classification task.