Jackson Cook

CS4641 – Machine Learning

Assignment 3: Unsupervised Learning

Classification Problems/Datasets

Vehicle:

This dataset was collected with the goal of classifying a vehicle by its silhouette as one of four types. By extracting a variety of scale independent features, the vehicle would be classified as OPEL, SAAB, BUS, or VAN. Two sets of 60 images were taken for each vehicle covering a full 360 degrees of rotation. This dataset contains 846 instances and each has 19 attributes.

Vehicle is interesting because vehicle identification is practical in life, especially in law enforcement. Classification will be very difficult as geometry of vehicles of all types tend to follow a similar trend. The VAN and BUS are expected to be easily distinguished between each other and amongst the OPEL and SAAB. The 4 OPEL and SAAB are similar in geometry and will likely be more difficult to classify. Overfitting is a major concern because of this, as many geometric features will probably be shared between all four vehicle types. Boosting will probably be very valuable on the dataset as it seems some of the instances may be much more easily classified than others.

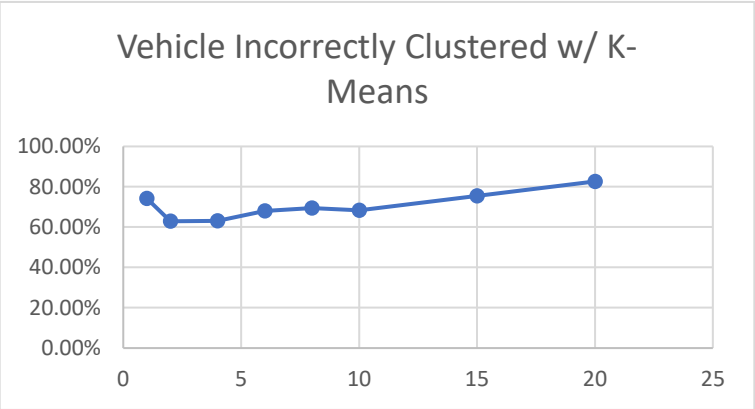
Semeion:

This dataset has the results of a study where handwritten digits from approximately 80 people were scanned, expanded into a 16 pixel by 16 pixel square, and given a Boolean for each pixel in the box if the pigment was at a certain threshold. There are 1593 instances in this dataset with 256 attributes each.

Semeion is valuable as character recognition is extremely useful in many fields especially in translation from handwritten to digital. Correct classification of numbers and in the future letters can provide tremendous utility to many people. Issues to expect with classification will likely be linked to peoples' different handwriting which varies from person to person. The dataset is also quite small given how many digits there are and how many instances per digit there are for the algorithm to learn. Overfitting may be useful when classifying this data, so I expect the neural network to excel on this dataset.

Part 1: *Run the clustering algorithms on the data sets and describe what you see.*

Vehicle K-Means

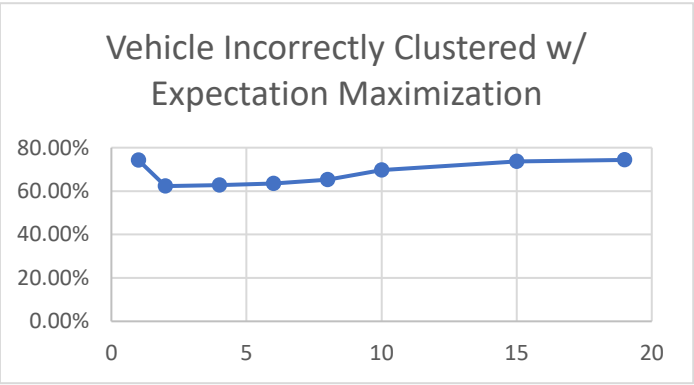


# of Clusters	Incorrectly Clustered	Within Cluster Sum of Squared Errors	# of Iterations
1	74.23%	578.2915013	1
2	62.88%	315.5856735	9
4	63.00%	223.5402089	8
6	67.97%	190.0413781	31
8	69.39%	167.881328	23
10	68.32%	153.6420063	24
15	75.53%	120.8481737	19
20	82.62%	103.4969026	18

Figure V1: Cluster accuracy graph and table over varying iterations also showing error and iterations for K-Means

I was expecting the optimal number of clusters to be 4 since the number of classifications for this data set was 4. The lowest incorrectly clustered percentage was 62.88% on 2 clusters. As can be seen in the graph showing the incorrectly clustered percentage, when the number of clusters increased, after reaching 4, the clustering algorithm began to overfit and began to incorrectly classify data quickly. The sum of squared errors decreased as the numbers of clusters increased. This is due to each cluster defining a smaller number of points in the space as more clusters formed, leading to each cluster being responsible for a smaller amount of the state space. The number of iterations grew as the cluster quantity grew due to more randomly generated centers of clusters needing to be relocated with each iteration, taking longer to reach the optimal location for each cluster.

Vehicle Expectation Maximization



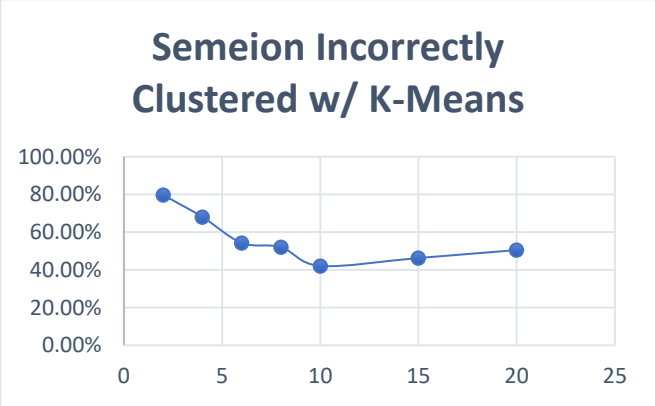
# of Clusters	Incorrectly Clustered	Log likelihood	# of Iterations
1	74.23%	-70.23094	2
2	62.29%	-64.06785	6
4	62.77%	-60.85744	20
6	63.48%	-57.94195	14
8	65.25%	-56.1425	1
10	69.62%	-55.0743	1
15	73.64%	-53.48275	1
19	74.35%	-52.79806	1

Figure V2: Cluster accuracy graph and table over varying iterations also showing error and iterations for Expectation Maximization

Once again my expectation was for the best number of clusters to be 4, but it was 2 again. This may be due to the clusters including many of the possible classifications but calling the cluster one of the classifications, getting some good

information on two of the classifications while having no chance of classifying the other two. The best incorrectly clustered was 62.29%, which is better than chance by about 13%, but is not great. Like in the previous analysis with K-Means, the accuracy once again fell off after 4 clusters, but it was not as bad due to expectation maximization clusters having the ability to contain more than one classifications. The iterations decreased as the number of clusters increased. This is also due to how expectation maximization clusters can include more than one classification, allowing for the initially generated clusters to account for all of the data.

Semeion K-Means

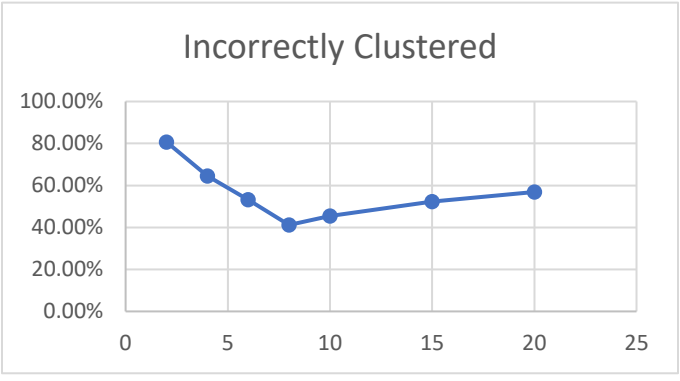


# of Clusters	Incorrectly Clustered	Within Cluster Sum of Squared Errors	# of Iterations
2	79.72%	80401.31597	38
4	68.05%	74543.57269	14
6	54.30%	71143.26894	35
8	52.10%	68469.29019	25
10	42.12%	66561.79142	24
15	46.33%	62796.70925	21
20	50.53%	60957.97522	14

Figure S1: Cluster accuracy graph and table over varying iterations also showing error and iterations for K-Means

For the Semeion dataset, I expected the optimal number of clusters to be 10, and it was the case. The Semeion dataset was much better described by the K-Means than the Vehicle dataset. This is likely due to the larger number of instances giving the learner more time to learn. However, just like with the Vehicle dataset, as the number of clusters increased beyond the number of classifications, the incorrectly clustered percentage increased. We also see the same behavior for the sum of squared errors and the number of iterations as we saw in the vehicle dataset.

Semeion Expectation Maximization



# of Clusters	Incorrectly Clustered	Log likelihood	# of Iterations
2	80.73%	-144.57856	3
4	64.47%	-124.01782	1
6	53.17%	-110.48799	1
8	41.18%	-92.80918	1
10	45.45%	-86.07861	1
15	52.35%	-73.48601	2
20	56.94%	-69.90387	2

Figure S2: Cluster accuracy graph and table over varying iterations also showing error and iterations for Expectation Maximization

This time the optimal number of clusters seemed to be 8 by a significant 4%. This is probably due to the ability of expectation maximization to cluster more than one classification in each cluster and while not having the ability to have a cluster for each classification still having better accuracy on the 8. The clustering accuracy once again decreases as the number of iterations increases past the optimal value which is near the number of possible classifications (8). The iterations are also very low and the error decreases with increasing clusters.

Part 2: *Apply the dimensionality reduction algorithms to the two datasets and describe what you see.*

PCA

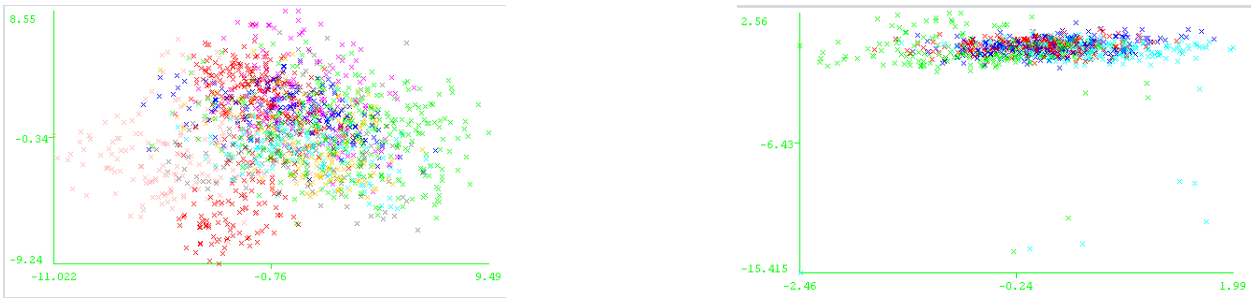


Image B1: Semeion on the left and Vehicle on the right showing PCA

PCA was able to break up the data very on Semeion very effectively (shown on the left). There are clear groupings of the data; for example, pink is on the left, light red on the bottom left, green on the right, etc. There are some regions which are harder to see in the center. I expected PCA to perform quite well, as given the knowledge of the domain. The filter on the perpendicular variance would allow small deviations in the writing of the numbers to be ignored.

PCA performed as expected on the Vehicle dataset. It was easily able to distinguish the Van (in green) and the Bus (in light blue), but struggled to distinguish between the two harder cases (in red and blue, falling in the middle of the grouping).

Randomized Projection

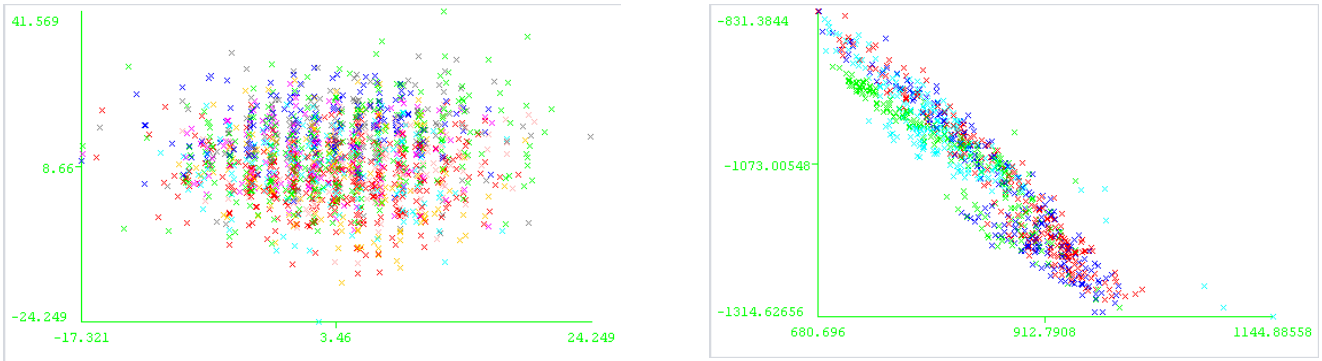


Image B2: Semeion on the left and Vehicle on the right showing Randomized Projection

Randomized Projection on the Semeion dataset is very unclear when I look at it. Too many of the classifications are too close together for me to get a good picture of the grouping. The groupings got slightly more discernable at higher numbers of dimensions and the run time for the algorithm increased with the higher numbers of dimensions as well.

Randomized Projection performed very comparably to PCA on the Vehicle dataset and yielded a graph that very closely resembles that of the PCA. There is still clear separation between the Van (green) and the Bus (light blue), and much confusion between the more difficult cases in the blue and red.

Auto Encoder

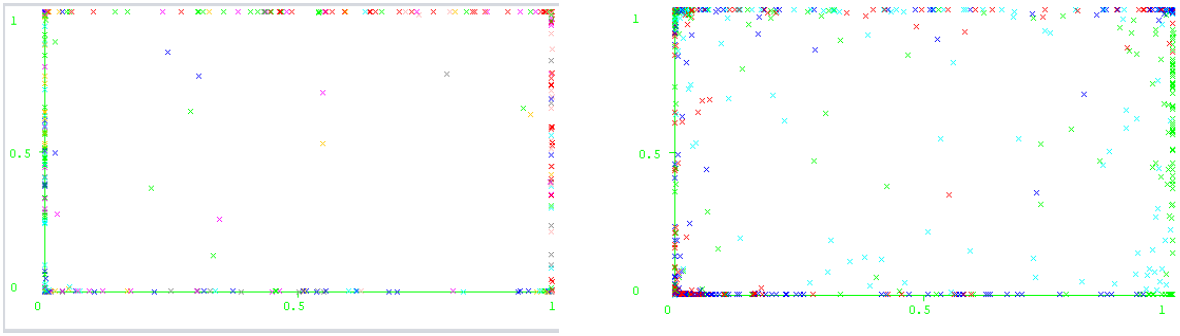


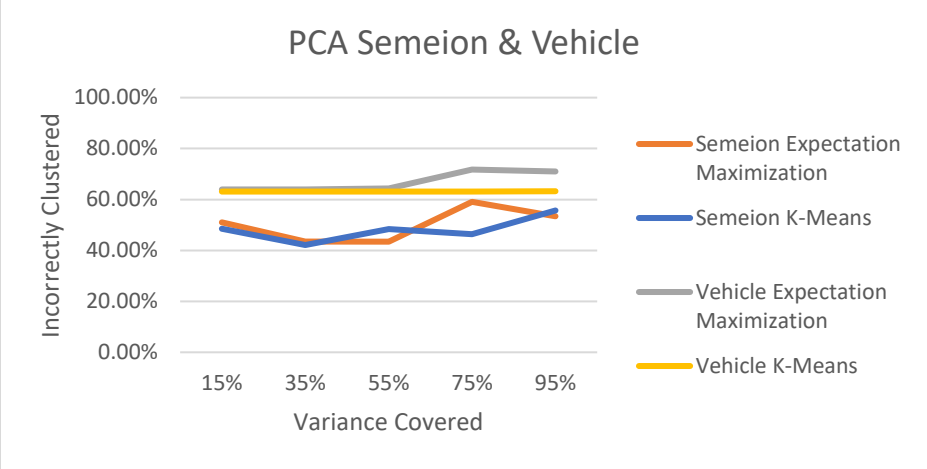
Image B3: Semeion on the left and Vehicle on the right showing Auto Encoding

The Auto Encoder like the Randomized Projection is very difficult to see differences on. For the Auto Encoder, since it uses a multi-layer perception with likely a threshold, most of the cases line the outside and create a sort of box of the data when projected to 2D. The Auto Encoder seems to have performed the worst of all of them but not much worse than Randomized Projection. Since the data is so complex and large, I may have needed a larger network than the 6 layer I used, but due to the time cost, I was unable to use too many.

For the Vehicle dataset, the Auto Encoder seems to have done pretty comparably to the others in terms of keeping the green and light blue apart while having the reds and blues still in groups together, although the groups of red and blue seem to be tighter for the Auto Encoder than the others. This means the algorithm was able to maintain important and distinct differences to keep the green and light blue apart but also may have lost some important variance between red and blue.

Part 3: *Reproduce your clustering experiments, but on the data after you've run dimensionality reduction on it.*

PCA



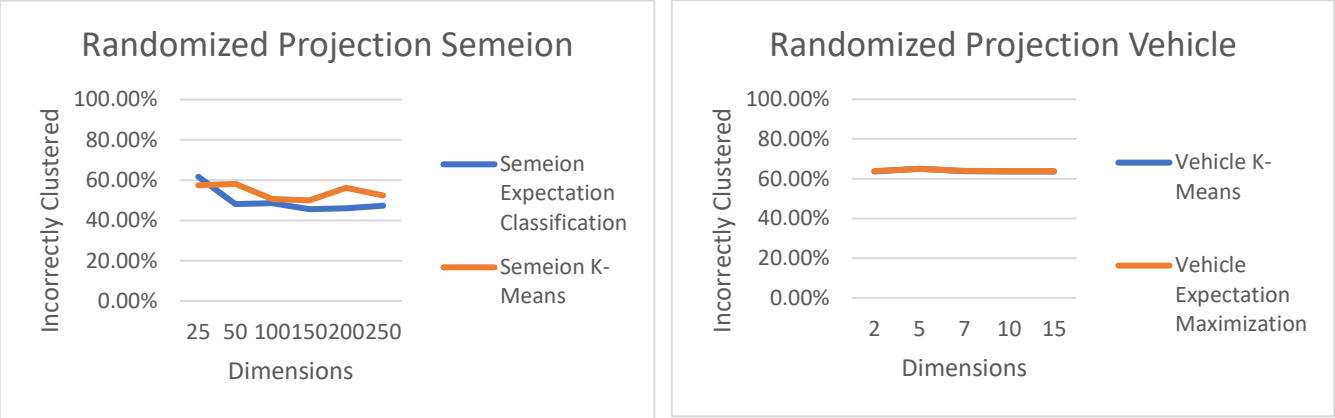
Graph B1: Graph showing incorrectly clustered percentage vs variance covered by PCA for Expectation Maximization and K-Means

For the Semeion dataset, PCA helped the clustering algorithm the most of all of the dimension reduction algorithms and performed very comparably to the accuracy of the dataset before dimensional reduction. It performed the best with a variance covered of 35% which reduced the number of attributes to 9. There is no discernable difference Expectation Maximization and K-means on this reduced data. From dataset knowledge, I believe PCA would be able to filter out minor differences between the writing of the numbers, allowing for better clustering. Large differences may be causing issues between differentiating difficult classifications such as "1" & "7" as was experienced before. PCA reduced data could be used to replicate the actual data for faster processing time of clustering algorithms.

For the Vehicle Dataset, PCA performed comparable to the other dimension reduction algorithms, having little to no effect on the incorrectly clustered percentage. The best performance was with a variance covered of 15%. There was very little change as the variance covered varied for both the Expectation Maximization and K-Means algorithms. The tendency of PCA to filter out slight differences may have caused it to ignore key aspects of the dataset given the complexity of clustering for this dataset. PCA reduced data could be used to replicate the actual data for faster processing time of clustering algorithms.

Randomized Projections

For Semeion, I used larger dimensions due to its already large dimension of attributes compared to the Vehicle dataset.

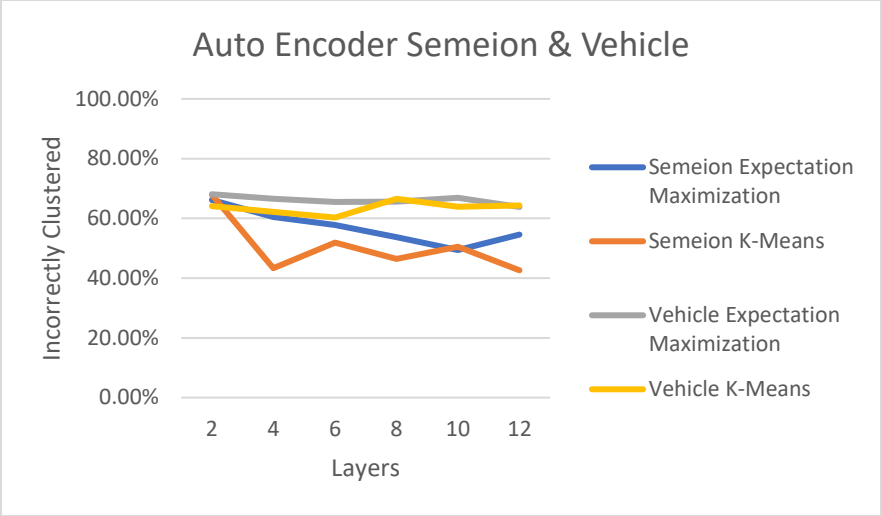


Graph B2: Graph showing incorrectly clustered percentage vs variance covered by RP for Expectation Maximization and K-Means

For the Semeion dataset, Randomized Projection had little effect on the performance of the clustering algorithms relative to the performance of the algorithms before dimensional reduction (slightly worse). I saw that as I increased the number of dimensions for the randomized projections, the clustering continued to improve. This occurred even when raising the dimension of the data. This is likely due to more of the data being preserved, defeating the purpose of dimensional reduction. However, the comparable performance of the dimensional reduction algorithm signals that the reduced dimensional data can be used for quicker run times and comparable results.

For the Vehicle dataset, once again the dimensional reduction had little effect on the performance of the clustering. This is good as it allows for lower dimension data storage, allowing for quicker run times although run time was not an issue with a dataset as small as that of the Vehicle dataset.

Auto Encoder

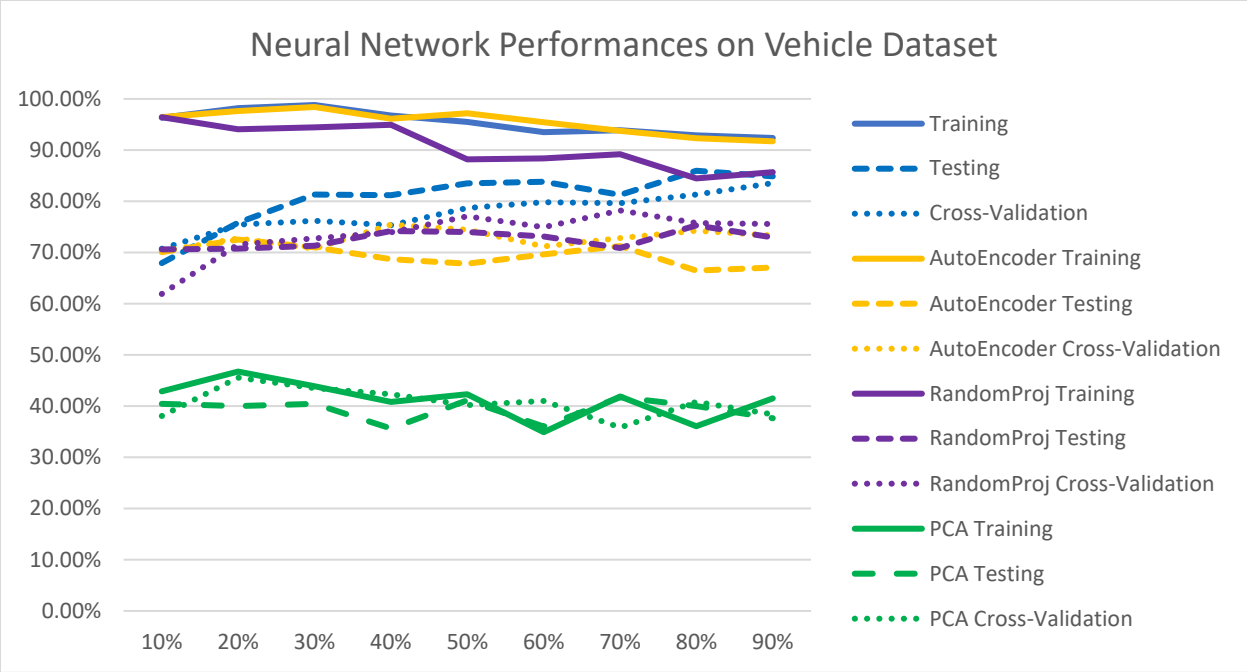


Graph B3: Graph showing incorrectly clustered percentage vs variance covered by AE for Expectation Maximization and K-Means

For the Semeion dataset, the Auto Encoder performed the best with 4 layers. The Auto Encoder also took a very long time to run especially at higher layers due to its being a multilayer perceptron and the size of the dataset. Once again, it seems like the dimensional reduction did not have negative effects on the correctness of clustering.

For the Vehicle dataset, the Auto Encoder performed best at 6 layers. It performed about 4% worse than the other two dimensional reduction algorithms, but is still very similar in clustering accuracy to the non-dimensionally reduced clustering accuracy. The slow performance of the Auto Encoder was not experienced on this dataset up to 12 layers due to the small size of the dataset.

Part 4: *Apply the dimensionality reduction algorithms to one of your datasets from assignment #1 (if you've reused the datasets from assignment #1 to do experiments 1-3 above then you've already done this) and rerun your neural network learner on the newly projected data.*



Graph V1: Graphs of each dimensionality reduction algorithm training, testing, and cross validation accuracies w/ neural network

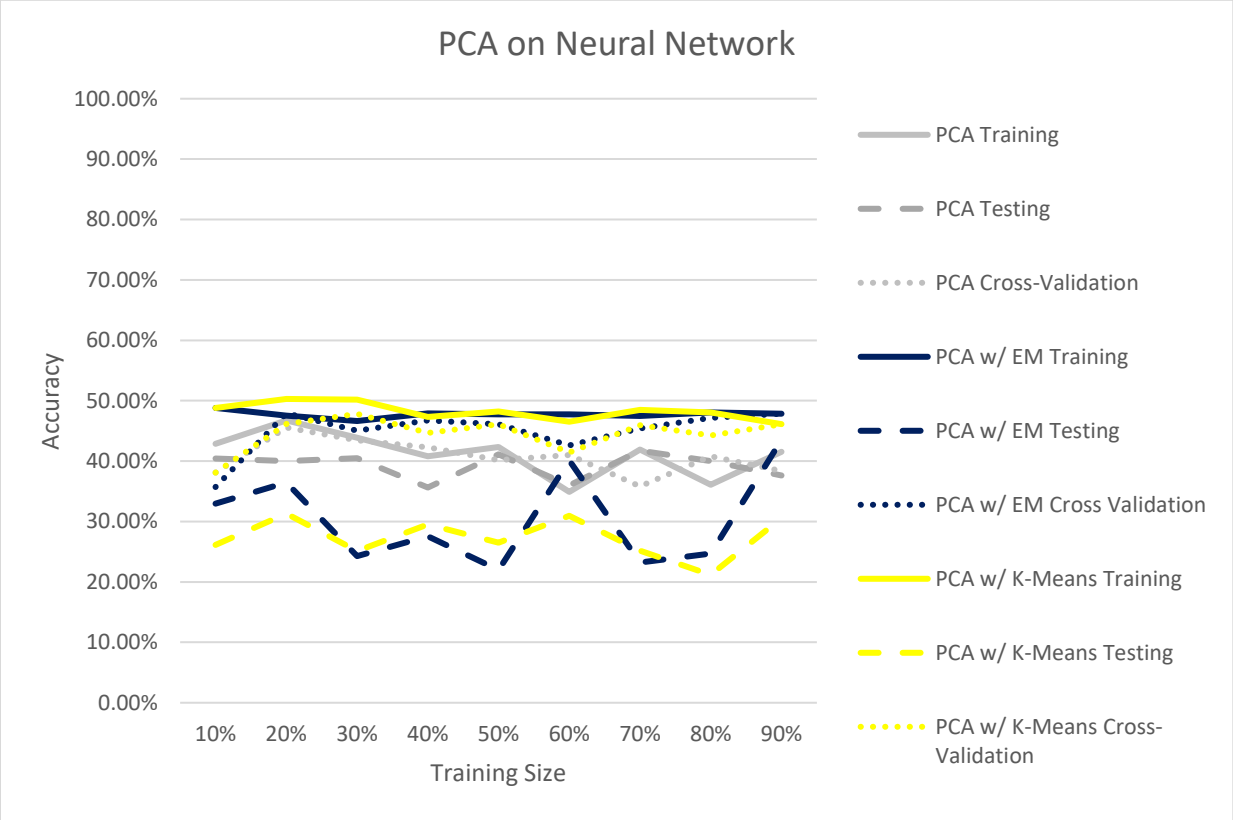
Drawing knowledge from project one, I used a learning rate of 0.1, a momentum of 0.2, and 1000 epochs for the neural networks as those were the values which yielded the best testing accuracy. Also, I chose the variance covered for the vehicle dataset to be 15% for PCA, the dimensions to be 2 for Random Projections, and the layers to be 6 for Auto Encoders based off of results from part 3 of this homework. PCA performed significantly worse than the rest of the dimensional reduction algorithms. This may be due to the low variance covered I chose of 15 percent reducing the attributes of the dataset from 19 to 2. Auto Encoders and Random Projections performed similarly with Randomized Projections coming out on top slightly, both with testing accuracies dancing around 70%. This leaves Randomized Projections about with about 10% lower testing accuracy than the testing accuracy of the original algorithm. The reduced dimension dataset was able to perform slightly comparably to the normal one. This could be helpful in terms of run time if the dataset had more instances. The same could be said for the Auto Encoder. If run time of the neural network was prioritized over the accuracy, these dimensional reduction algorithms could be used to reduce the complexity of the dataset before sticking it into the neural network. Also, the training accuracy of the Randomized Projections decreases quicker than the Auto Encoder, but still performed well on the testing set showing the power of generalization its reduction holds over the Auto Encoder.

The PCA may have been improved with a higher variance covered than the 15% I chose due to it being the best result in the clustering. The Auto Encoder may have been able to be improved by varying size of the hidden layers. The Randomized Projections may have been improved by increasing the dimension chosen since 2 is

fairly small and reduced the number of attributes to 3 which seemed to hinder PCA, but that may make the state space of the data too large for the neural network to easily grasp without overfitting to outliers. All of the dimensional reduction algorithms could obviously be improved by having more instances.

Part 5: *Apply the clustering algorithms to the same dataset to which you just applied the dimensionality reduction algorithms (you've probably already done this), treating the clusters as if they were new features. In other words, treat the clustering algorithms as if they were dimensionality reduction algorithms. Again, rerun your neural network learner on the newly projected data.*

PCA

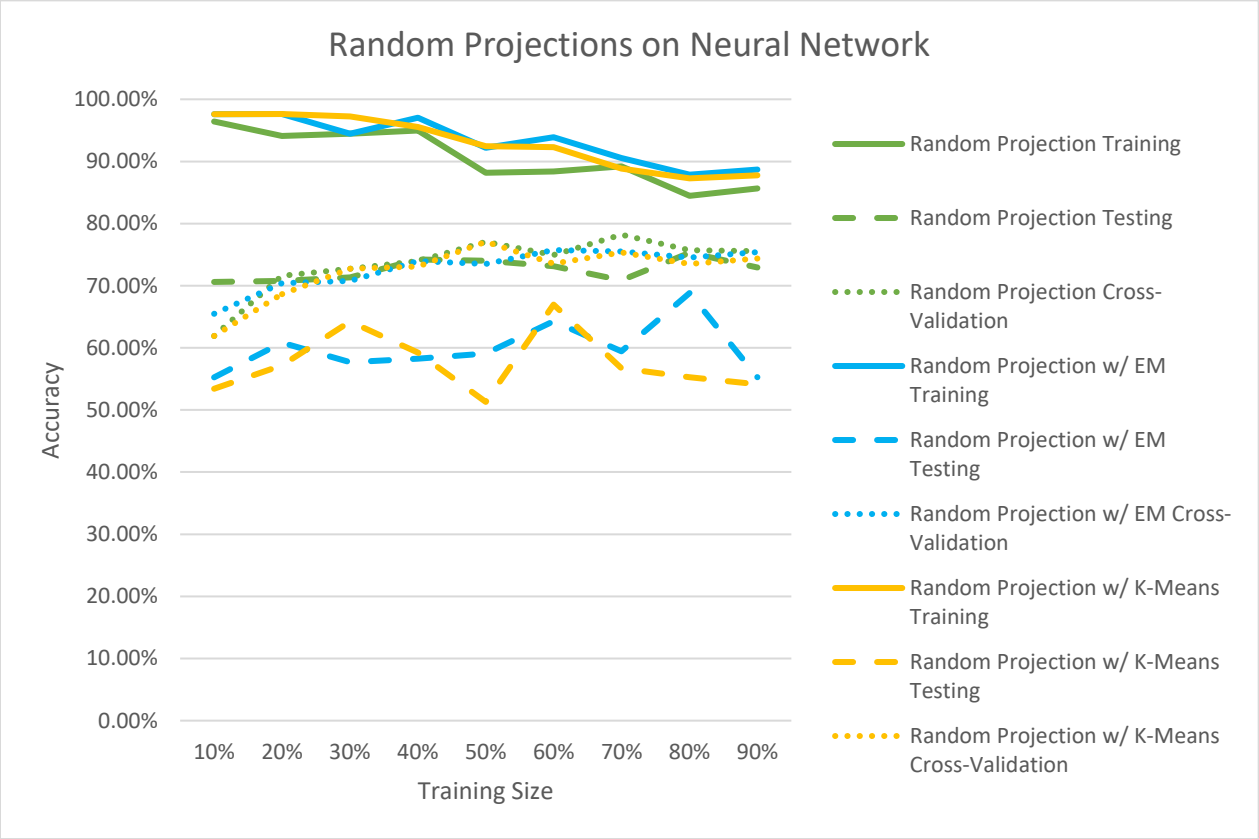


Graph P1: Graph of accuracy vs training size for different variations on a PCA transformed dataset input into a neural network with learning rate 0.1, momentum 0.2, and 1000 epochs. PCA done with 15% variance

On the testing set, the PCA transformed data performed much better than the PCAs with additional clustering added on. I believe this is because when the data was initially reduced in dimension, it changed the space of the classification. Adding in another algorithm made the classification with the neural network worse because the neural network then had to work in a further transformed space that was more difficult to classify. Also, the clustering algorithms did not perform very well in clustering the data as seen in parts 2 and 3, so they did not help but rather hurt the classification.

The neural network was forced to learn a smaller, badly clustered space. The neural networks ran on this data ran significantly quicker than on the others, likely due to the low variance covered I chose (15%) making the data very small.

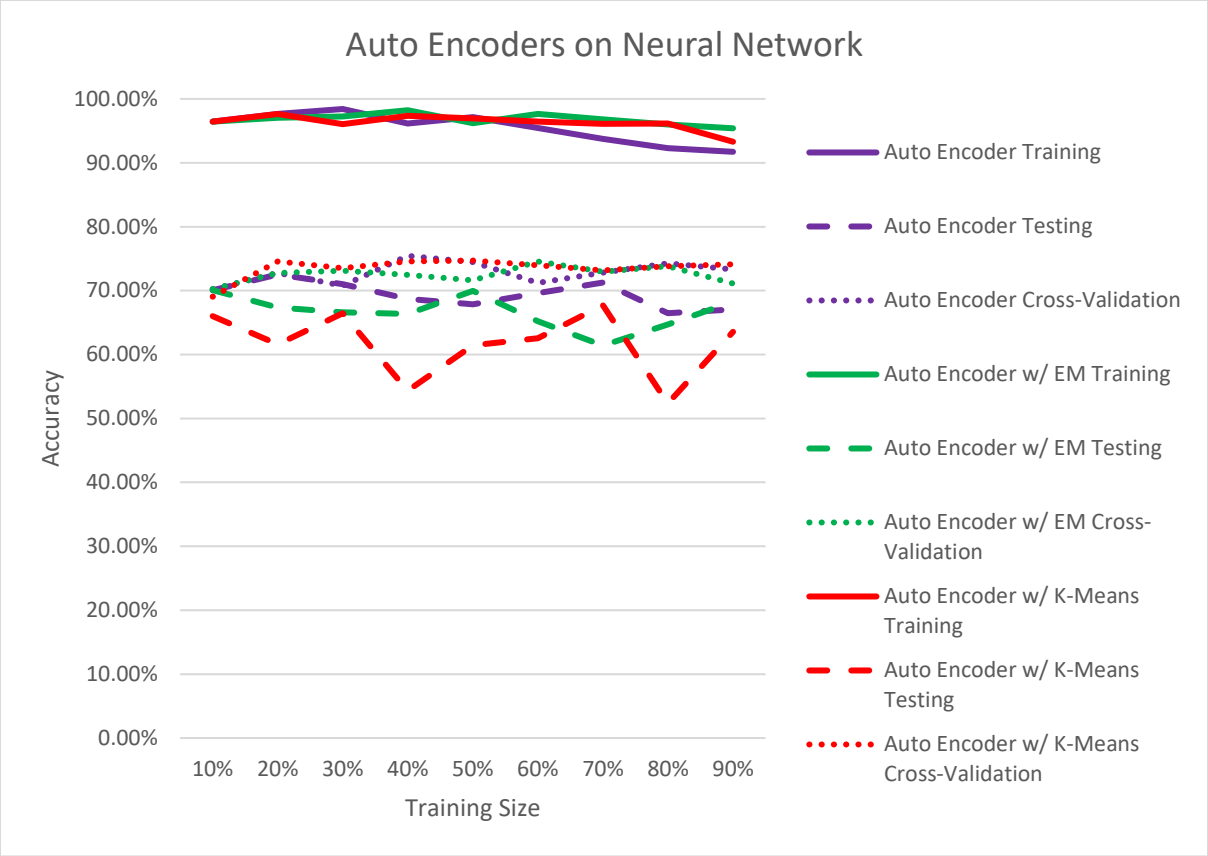
Random Projection



Graph P2: Graph of accuracy vs training size for different variations on a Random Projection transformed dataset input into a neural network with learning rate 0.1, momentum 0.2, and 1000 epochs. Random Projection done with 2 dimensions

The addition of clustering on random projections also did not help but hurt the learning of the neural network. This once again suggests that learning the clustered data has hurt the learner by feeding it already flawed data by over complicating the state space with unnecessary operations.

Auto Encoder



Graph P3: Graph of accuracy vs training size for different variations on an Auto Encoder transformed dataset input into a neural network with learning rate 0.1, momentum 0.2, and 1000 epochs. Auto Encoder done with 6 layers

The Auto Encoding + Expectation Maximization is the only time an added clustering algorithm came close to just using the auto encoder. As seen in part 2, the Auto Encoder did a pretty good job of differentiating between cases using its multi-layer perceptron. This fed in well to Expectation Maximization especially because of Expectation Maximization's ability to have a point shared between two clusters to help differentiate between the two most difficult cases for the Vehicle dataset.

Conclusion

All of the dimensional reduction algorithms can be helpful. I experienced on a few occasions their power to speed up the run time of the neural network/cross validation, and as can be seen in part 4 for my vehicle dataset, the Random Projection transformed data was able to make comparable results to un transformed data but much quicker. All of my results could obviously be improved if I had more data, but I believe good results were seen. Expectation Maximization seemed to perform better on the Vehicle dataset because it never had to guess between the two very similar classifications that dataset has (the SAAB and the OPEL). The Auto Encoder was the most costly in terms of time, followed by Random Projections.