Balkrishna Patel bpatel66 CS 4941

Assignment 3: Supervised Learning and Dimension Reduction

Voice-Classification Dataset (K-Means)

For my first dataset I will be using the same Voice-Classification Dataset that I used in assignment 1 because I am very interested to see how clustering and dimension reduction algorithms behave on it. I assume for K-Means will try and split the dataset into 2 clusters because of the 50-50 class split the dataset shows.

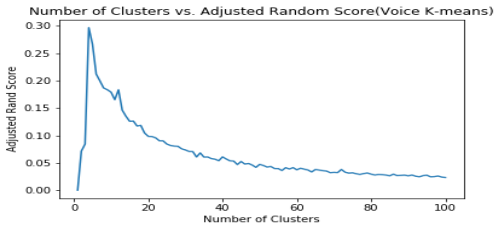

Figure 5: Number of Clusters vs. Adjusted Random Score (Voice K-Means)

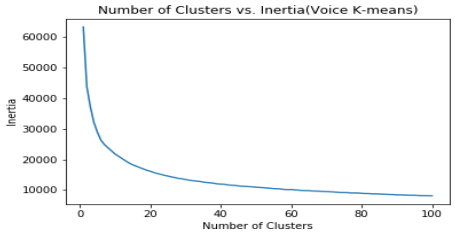

Figure 6: Number of Clusters vs. Inertia (Voice K-Means)

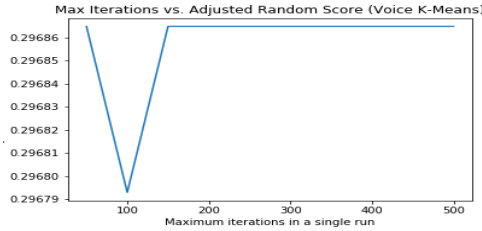

Figure 7: Max Iterations in Single Run vs. Adjusted Random Score (Voice K-

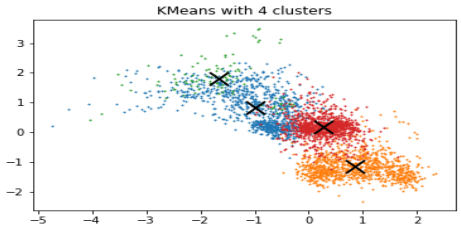

Figure 8: Visualization of the 4 Clusters (Voice K-Means)

The first algorithm I am going to experiment with on this dataset is K-Means Clustering. I will be using Adjusted Random Score as the performance metric for this algorithm. Adjusted Random Score is going to give a score 0.0 to 1.0 depending on how similar the two clusters are, 0.0 shows that they are very different and 1.0 shows that they are identical. We are trying to get the highest Random Score.

The first experiment I wanted to conduct was to see how many clusters were optimal my running K-Means on my dataset. If two clusters are enough that would mean my dataset is linearly separable which given this dataset I would assume it is. To figure out if my hypothesis is correct or not I ran an SVM without a kernel and found that my dataset is not linearly separable. So maybe three or four clusters should be enough, to figure out the optimal number of K-Clusters I varied the number of clusters and mapped the Adjusted Random Score. I also wanted to see how error was impacted as the number of clusters increased. I mapped the relations between the number of clusters vs adjusted random score and inertia (which is the sum of squared errors) in Figures 5 and 6 respectively.

From Figure 5 we can see that four is the optimal number of clusters. Also, from Figure 6 we can see as the number of clusters increases the error/inertia decreases almost exponentially, this decrease could be because as the number of clusters increases the distance to the center of the cluster from the data points is getting smaller.

The next thing I wanted to see is how many iterations in a single run did it take for there to convergence and I also wanted to see how the clusters looked, if graphed these two experiments/observations in Figures 7 and 8 respectively. From Figure 7 we can see that around 100 iterations has a Random Score of 0.29679 and are about 150 iterations the Random Score peaks at 0.29686. In Figure 8 we can see the four clusters, after seeing the visualization of the clusters and how the data points are laid out you can see how two clusters wasn't enough. With the final experiment concluded for K-means on the voice-classification dataset we saw that the random index reached a peak of about 0.30 at four cluster and took about 150 iterations to converge. It kind of looks like three cluster could work out as well if we are will to sacrifice some of the Adjusted Random Score.

Voice-Classification Dataset (Expectation Maximization)

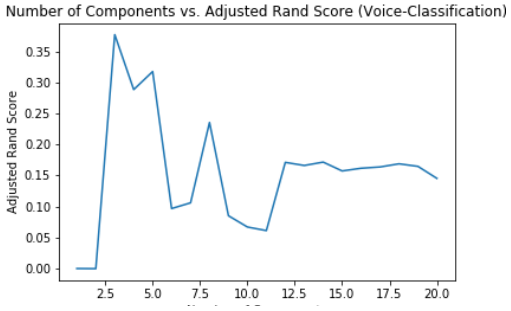

Figure 9: Number of Components vs. Adjusted Random Score (Voice EM)

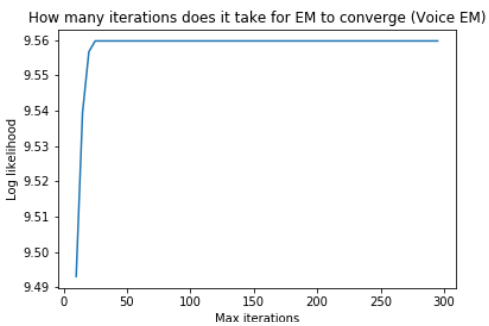

Figure 10: How many iterations does it take for EM to converge

The next algorithm I will be experimenting with is the Expectation Maximization algorithm. For this algorithm I will be using Sci-Kit Learn's Gaussian Mixture implementation. To begin my experimenting, like what I did for K-Means I want to see how varying the number of component s impacts the Adjusted Random Score. You can see the relationship

between number of components and Adjusted Random Score in Figure 9.

From Figure 9 we can see that three components give us the maximum Adjusted Random Score, this is consistent with our finding in K-Means. The main difference is that EM has a slightly higher Random Index meaning the EM is doing a better job of clustering. Because EM uses elliptical clusters the data from the dataset may be better modeled under EM and because of EM's soft clustering (where instead of a datapoint being assigned to a specific cluster it is given a likelihood of which cluster it most fits with) EM is able to model data on the edges better. In Figure 10 we can see it take about 50 iterations of EM to converge.

Voice-Classification Dataset (Expectation Maximization)

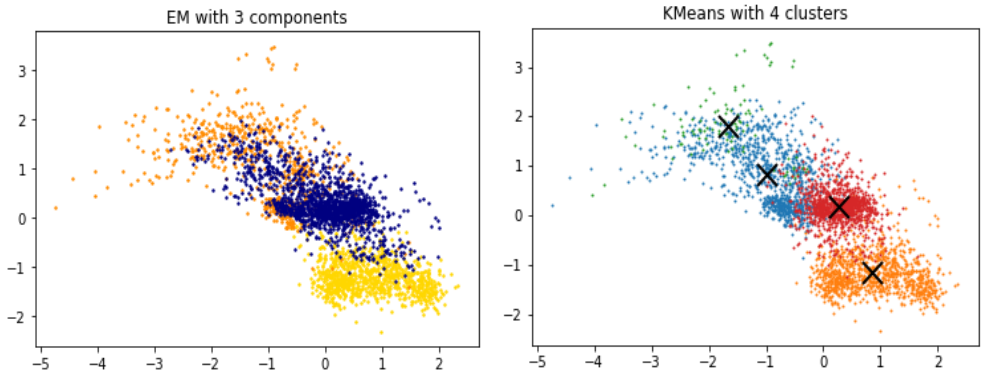

Figure 11: Cluster Visualization for K-Means and EM for Voice Dataset

From Figure 11 we can see how EM was able to get a higher Random Index than K-Means, the Yellow Cluster in the EM graph and the Orange Cluster in the K-Means graphs are almost if not fully identical the only real difference comes from how the Purple Cluster in EM was able to take the place of the Red and Blue Clusters in K-Means. This is probably due the elliptical nature of EM allowing points further from the center allowing for one fewer cluster. For this dataset EM seems to be superior but K-Means is probably suited more for datasets with datapoints that are linearly separable.

Voice-Classification Dataset (Principal Component Analysis (PCA))

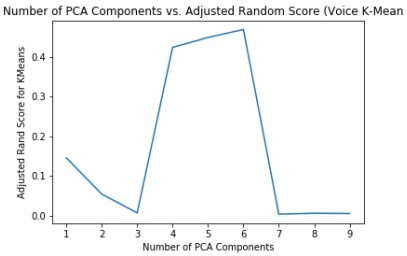

Figure 12: Number of PCA Components vs. K-Means ARS (Voice)

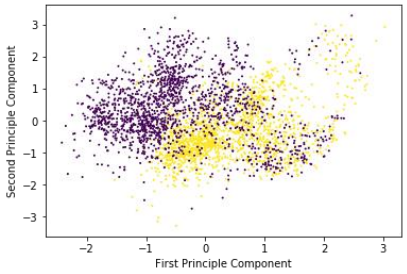

Figure 13: K-Means with 1st and 2nd PCA

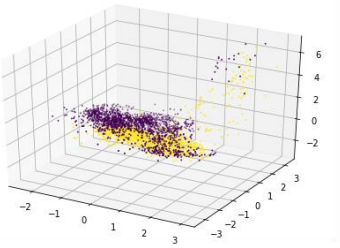

Figure 14: K-Means with 1st, 2nd, and 3rd PCA

The next algorithm I will be experimenting with is the first of four dimension reduction algorithms I will look at, it is Principle Component Analysis or PCA. PCA is a linear dimensionality reduction that uses SVD (Singular Value Decomposition) of data points from the dataset to project it to a lower dimensional space. First, I wanted to see the how many features are needed to cover the variance in the dataset. I found that the first two features covered 57.09% of the variance and the first three covered 68.00% of the data and finally I found that the first nine features covered 94.08% of the variance. After the first nine features there was improvement in coverage but not as substantial of improvements as the first nine features showed. So, the remain features will be

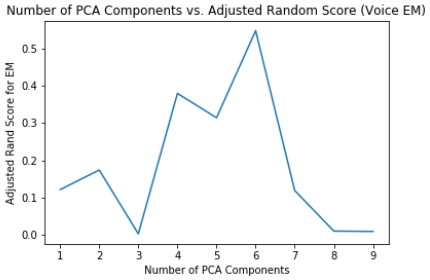

Figure 15: Number of PCA Components vs. EM ARS (Voice)

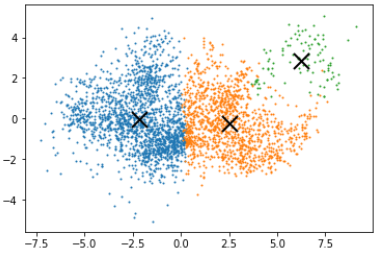

Figure 16: K-Means with 3 Clusters and 6 PCA

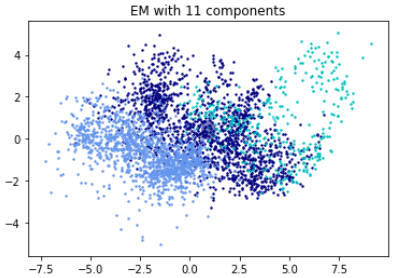

Figure 17: EM with 3 Clusters and 6 PCA

retracted. Next, I wanted to see Random Index for K-Means was impacted as we increased the number of PCA features up to nine. I findings are shown in Figure 12.

From Figure 12 we can see that initially from one to three components the ARS declines to zero. We reach a peak score at six PCA components. After six components the graph shows a steep decline back down to zero, this is probably because PCA is suffering from the curse of dimensionality. Adding the seventh component doesn't add much information based on the graph, and because of this K-Means is suffering. In Figures 13 and 14 I have shown what K-Means with 3 clusters looks like with 2nd and 3rd PCA. We see that it is not possible to separate the datapoints into 3 clusters because of how much they are mixed together. When we look at 3rd PCA we can see that the Yellow points are basically sandwiched in between the Purple datapoints.

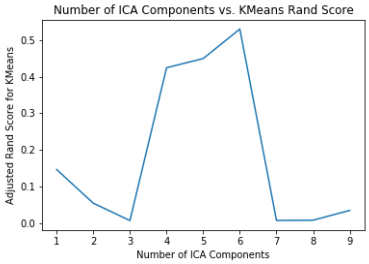

Figure 18: Number of ICA components vs. K-Means Rand Score (Voice)

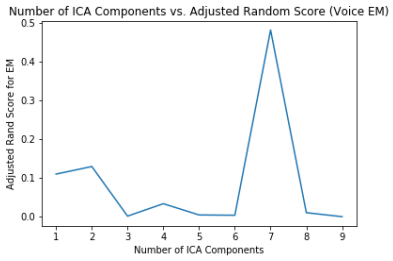

Figure 19: Number of ICA components vs. EM Rand Score (Voice)

Now let's look at how the number of PCA components impacts the Adjusted Random score of EM. The graph of the impact on ARS by varying PCA components is shown in Figure 15.

In Figure 15 we can see that six PCA components still gives us the highest ARS and that EM like K-Means suffers from the curse of dimensionality. With what we can see we only need to look at the first size features. So, after feature reduction we are only looking at less than a third of the features we have available to us in the dataset.

When we look at the clustering done by K-Means (Figure 16) and EM (Figure 17) with PCA applied we can see K-Means is outperforming EM simply by inspection by looking how well the clusters are separated. A reason why EM might be underperforming is because EM assume that data is coming from a Gaussian distribution and K-Means looks at a cluster and looks to make it more homogeneous thus yielding a higher Random Score.

In conclusion of PCA: six PCA Components with K-Means out performs EM by producing better clusters. Thus, if I am using PCA to do dimension reduction I would favor K-Means.

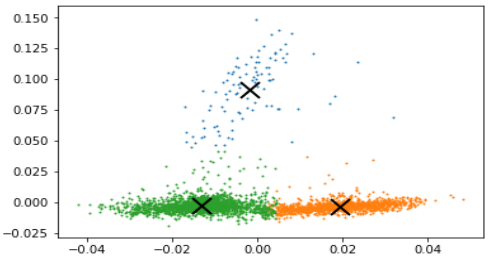

Figure 20: Voice K-Means Cluster with 3 Clusters and 6 ICA

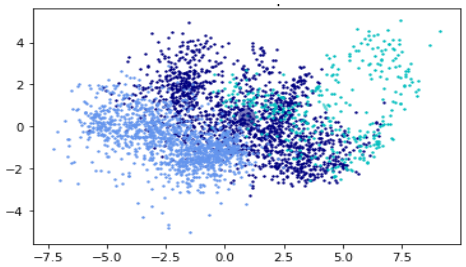

Figure 21: Voice EM Clustering with 3 clusters and 7 ICA

Voice-Classification Dataset (ICA)

The next dimension reduction algorithm I want to experiment with is ICA. To help me with my experimenting I will be using Sci-Kit Learn's FastICA library. The first thing I wanted to see was how varying the number of ICA Components would impact the Random Score of K-Means and EM. I have graphed the relations in Figure 18 and 19 respectively.

In Figure 18, we see that same thing we saw in Figure 12, the graphs are identical. In Figure 18 we see that the Random Score goes up after the ICA components get past 3 and it continues to increase until it get to 6 ICA components then it declines rapidly.

In Figure 19 we see some thing interesting. From 1 component to 2 the score is around 0.12 and after 2 it declines to around 0 after which at 7 the score peaks at .48 and then goes back down to 0 after.

Now that we know the optimal number of ICA component for K-Means and EM lets see what the clustering looks like to see which clustering algorithm performs best with this dimension reduction algorithm. I have graphed the clusters in Figures 20 and 21 for K-Means and EM respectively. Also, I am maintaining the three clusters as we used previously. From the 2 figures we can see that K-Means seems to again perform better than EM. We could've made this assumption before even looking at the clusters as the Rand Score of K-Mean was higher than that of EM.

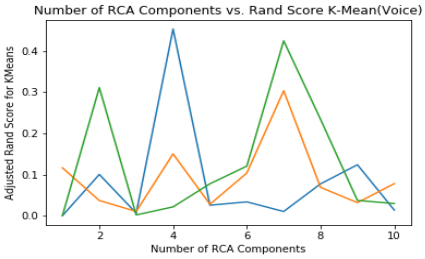

Figure 22: RCA Components vs. Rand Score K-Means (Voice)

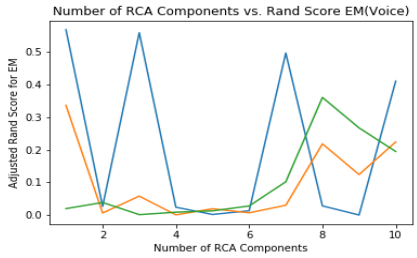

Figure 23: RCA Components vs. Rand Score EM (Voice)

Voice-Classification Dataset (Random Projection (RP))

The next algorithm I will look at is Random Projection. Random Projection or RP is an algorithm that reduces dimension by trading accuracy, it is more computationally efficient than the previous two algorithms. For the experimenting I will use Sci-Kit Learn's SparseRandomProjection library because of the flexibility it has with its parameters.

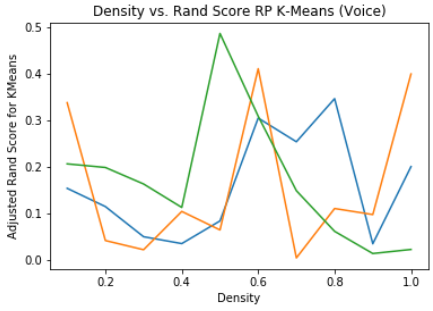

Figure 25: Density vs. Adjusted Random Score K-Means (Voice)

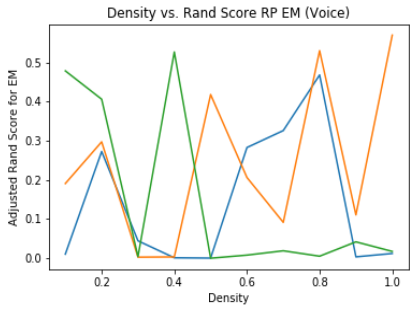

Figure 25: Density vs. Adjusted Random Scores EM (Voice)

The first thing I want to look at is again how the number of components created affect the Rand Score of K-Means and EM. I have graphed the relations in Figures 22 and 23 for K-Means and EM respectively.

In Figures 22 and 23 we can see the variation that came from different runs where RP was run on K-Means and EM. In Figure 22 we can see that the lows reach 0.0 and the highest peak is at around 0.45 for K-Means. As for Figure 23 we can see lows reach 0.0 and the highest peak reaches 0.52 or 0.54 for EM. These inconsistencies make RP less practical to use for important real-world applications. Before we disregard RP altogether maybe we can tweak a parameter to make it more stable. Next, I will look at what happens to the two clustering algorithms when I tweak the "Density" parameter. Figures 24 and 25 show the relationship between the varying density and Rand score for both the K-Means and EM.

Voice-Classification Dataset (Autoencoders (AE))

The final algorithm I will look at is called Autoencoders. Autoencoders or AE are neural nets that try to learn identity functions. For this algorithm I will tweak parameters like how many features the AE can produce and the number of nodes in the hidden layer to see If we can get some improvements in Adjusted Random Score. Figure 26 shows how the number of features created impacted the Adjusted Rand Score for K-Mean. From Figure 26 we can with the scale shown at the top of the graph the change is almost 0 (0.125 to be exact.) What this tells us is that the AE is generated features that have no correlation to the original features. In other words, its generating almost random features. Now let's see if we change the number of nodes in the hidden layer. Figure 27 show the how the EM's Adjusted Random Score changes with when the number of nodes in the hidden layer changes.

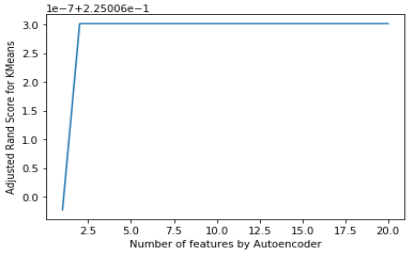

Figure 27: Number of Created Features vs. K-Mean Rand Score (Voice)

From Figure 27 we can see that 3 nodes seems to be the optimal number of nodes in the hidden layer as it achieves the highest Random Score (0.175).

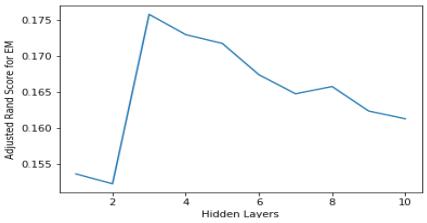

Figure 27: Number of Nodes in Hidden Layer vs. Adjusted Random Score EM (Voice)

For the features generated by the AE we achieved the lowest Random Score of 0.125 so our initial observation that the AE was generating features at almost random is pretty accurate. After tweaking the number of nodes in the hidden layer we were able to get the Rand Score up to 0.175 which was an improvement but not as nice as the scores we got previously.

Voice-Classification Dataset ANN on features created by PCA, ICA, RP, AE

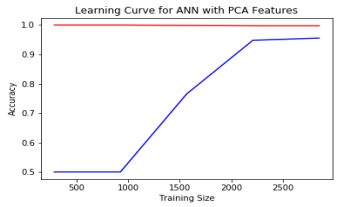

Figure 8: ANN Learning Curve w/ PCA Features

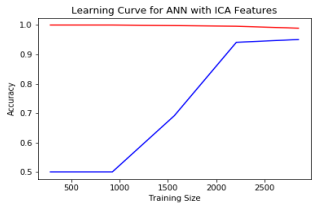

Figure 29: ANN Learning Curve w/ ICA Features

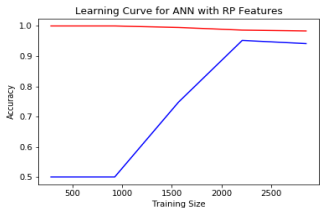

Figure 30: ANN Learning Curve w/ RP Features

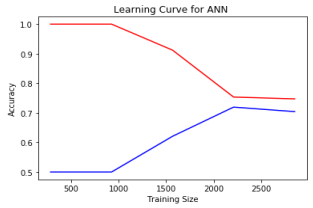

Figure 31: ANN Learning Curve w/ AE Features

Now we will look to see if the features created by the previous 4 algorithms yield any improvements over our results from Assignment 1. To see if we got any improvements I will plot the learning curves for the Neural Net with the features created by the previous algorithms. I have graphed the ANN learning Curve in Figures 28, 29, 30, 31 for PCA, ICA, RP, and AE respectively. From the figures we can see that accuracies for 2 of them are converging to our previous baseline from assignment 1 (95.65%). PCA and ICA are converging to out baseline and it looks like ICA may even pass it if we had enough data. The interesting curves come from RP which get to out baseline but then begins to decline, this shows that RP's Feature are not preforming the too well. RP is preforming

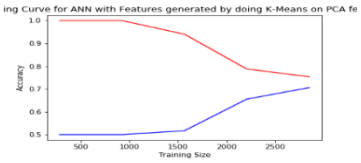

Figure 32: ANN Learning Curve w/ Features Generated by K-Means (PCA)

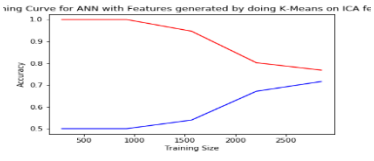

Figure 33: ANN Learning Curve w/ Features Generated by K-Means (ICA)

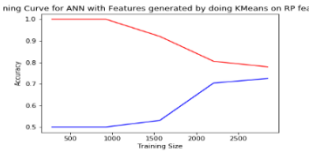

Figure 34: Ann Learning Curve w/ Features Generated by K-Means (RP)

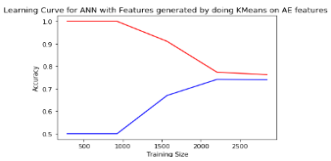

Figure 35: ANN Learning Curve w/ Features Generated by K-Means (AE)

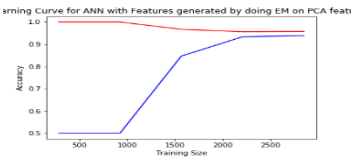

Figure 36: ANN Learning Curve w/ Features Generated by EM (PCA)

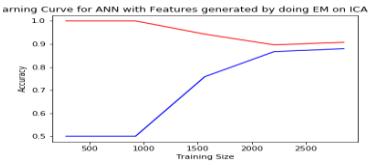

Figure 37: ANN Learning Curve w/ Features Generated by EM (ICA)

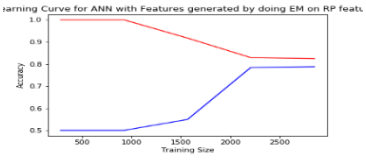

Figure 38: ANN Learning Curve w/ Features Generated by EM (RP)

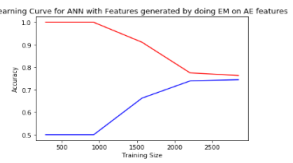

Figure 39: ANN Learning Curve w/ Features Generated by EM (AE)

much better than AE though, we see that AE never get to the baseline and event the training set's accuracy declines, this is due to the fact that AE is basically creating random not useful features.

Voice-Classification Dataset ANN on features created by K-Means, and EM

Lastly for the voice-classification dataset we are going to take a look at the learning curves of the ANN with features generated by K-Means and EM with all the Dimension Reduction algorithms. I've graphed the Learning curves for K-Means with PCA, ICA, RP, AE in Figures 32, 33, 34, 35 respectively. I've also graphed the learning curves from EM with PCA, ICA, RP, AE in Figures 36, 37, 38, 39.

From Figures 32-35 we can see that the accuracies are much lower across the board. This shows that using K-Means distance metric is a weakness to the ANN. Also now we only 3 weak features (clusters). Looking at Figures 36 – 39 we can see that even these accuracies are done can we can draw the same conclusion for EM that we drew from K-Means, the ANN is limited by have only 3 features and 3 weak features at that.

Image-Segmentation Dataset (Artificial Neural Net Analysis)

For my second dataset I chose not to use my Pulsar Dataset from Assignment 1 because I didn't see that it would behave well with clustering, because of how low of a percentage of the data was actually labeled as a pulsar. Also because the dataset showed that there were quite a few false positives I don't think clustering would've been able to cluster the 2 classes adequately. For this assignment I am replacing my Pulsar Dataset with the Image-Segmentation Dataset. I chose this dataset because unlike the Voice and Pulsar datasets this dataset has 7 evenly distributed classes instead of 2 classes. Because I am using a new dataset I will provide an analysis of a Neural Net that I would've run had I chosen this dataset for assignment 1.

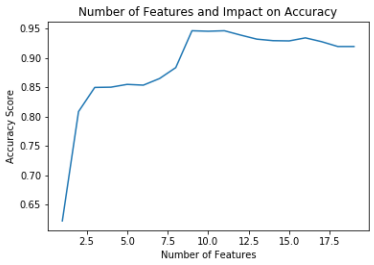

Figure 40: Number of Features vs. Accuracy (Image Segmentation)

In assignment 1 I started examining how increasing the number of features that were used impacted the accuracy of the model. For this I used looped through the number of features and ran a KNN model with 6 neighbors and graphed the accuracy scores against the number of features. My finding can be found in Figure 40. In the figure we can see the peak accuracy occurs at about 9 features after which the accuracy starts to decline. The decline could be caused by overfitting but I don't believe it as the decline is more steady like its going to normalize. Going forward for this Neural Net I will use 9 features.

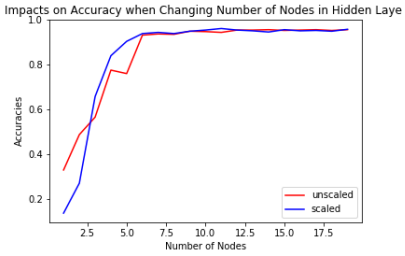

Figure 41: Number of Nodes in Hidden Layer vs. Accuracy (Image Segmentation)

Next, I looked at how increasing the number of nodes in the hidden layer would impact accuracy. My finding can be found in Figure 41. And for this I checked both the scaled and unscaled features to see if there was any performance gains from using the scaled features. From the figure we can see that the scaled features are ever so slightly better than the raw/unscaled features. We can also see that at about 10 nodes the graph starts to stabilize, so I will go with 10 nodes and 9 scaled features..

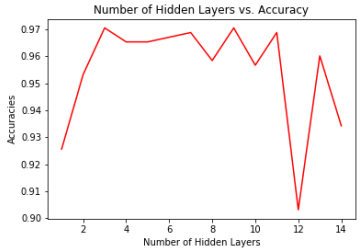

Figure 42: Number of Hidden Layers vs. Accuracy (Image Segmentation)

After I determined the number of nodes that would be optimal for the hidden layers the next logical step would be to determine how may hidden layers would be optimal. For this I increased the number of hidden layers from 1 to 14 all with 10 nodes in each and looked to see if how many would be sufficient. My finding can be found on Figure 42. From the figure we can see that 3 hidden layers yields the highest accuracy.

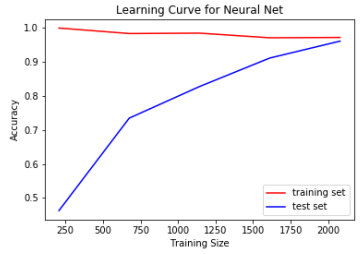

Figure 43: Learning Curve of ANN with 3 hidden layers w/ 10 nodes each

Now Its time to put all the information we gained together. Figure 43 shows the learning curve of a Neural Net run with 3 hidden layers with 10 nodes each. The max accuracy we were able to achieve with the dataset was 96.02% and it looks like if we had enough data the NN would've converged a little higher than 96%. Now its time to see how the dataset behaves with some clustering and dimension reduction algorithms.

Image-Segmentation (K-Means)

Like with the Voice-Classification Dataset I looked at how the number of clusters would impact the Random Index of K-Means. In Figure 44 and 45 you can see how the Adjusted Random Score and Inertia changed as the number of clusters increases respectively. From the graph we can wee that even though when the number of clusters is equal to the number of classes we have the ARS is high but at 8 clusters we reach our peak Score.

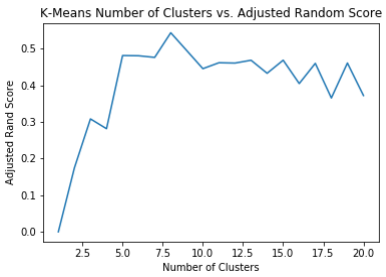

Figure 44: Number of Clusters vs. ARS (K-Means)

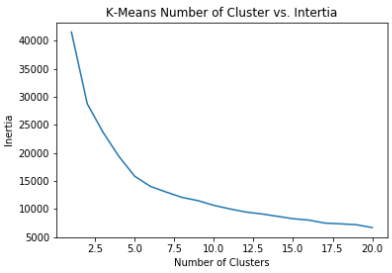

Figure 45: Number of Clusters vs. Inertia (K-Means)

Now look at what the data looks like with 8 clusters and for fun lets also see what 7 clusters looks like because that is the number of classes we have. You can see the 8-cluster visualization and the 7-cluster visualization in Figure 46 and 47 respectively. As we can see in the Figures with both 8 and 7 clusters we get relatively nice horizontal sections. We can see that for both figures that if you were to draw a horizontal link at y=0 you can see that part of the graph is super congested. At 7 clusters we can see that clusters are getting a bit more mixed up than with 8 clusters, this is how we get a higher Adjusted Random Score with 8 then 7.

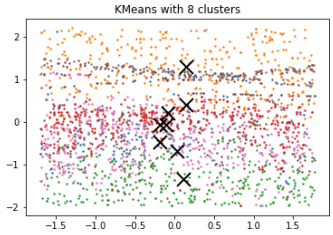

Figure 46: K-Means w/ 8 Clusters

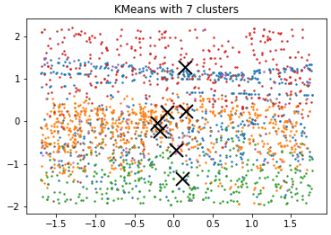

Figure 47: K- Means w/ 7 Clusters

The places we see a lot of datapoints clustered together are probably coming the fact that brick and concrete may look that same when looking at the numbers, so when we increase the number of clusters we get that K-Means finds it easier to differentiate between 2 similar images.

Another thing I wanted to see if like with the previous dataset does K-Means again converge after a max number of iterations? I have graphed the number iterations vs. ARS in one run in Figure 48. From the figure we can see that because the clusters are so close to one another K-Means doesn't have a chance to converge.

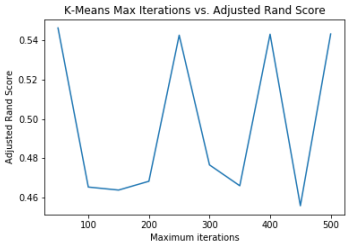

Figure 48: K-Means Max Iterations vs. ARS

To conclude K-Means here's what we found: 8 clusters gave us the best Random Index (just over 0.5) and it is difficult for K-Means to converge because of how close the clusters are to one another.

Image-Segmentation (Expectation Maximization)

Similar to how I did K-Means I am going to use the methodology I used for the previous datasets Expectation Maximization (EM) to see how this dataset behaves with EM. First let's take a look at how varying the number of components impacts the Random Index and Log Likelihood. I have graphed the 2 relationships in Figures 49 and 50 respectively.

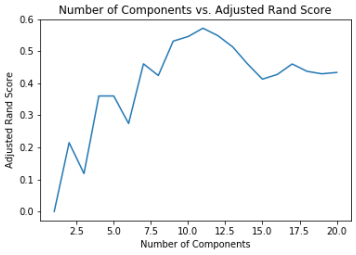

Figure 49: EM Number of Component vs. ARS

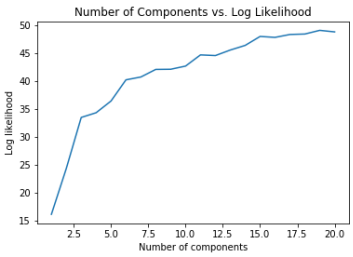

Figure 50: EM Number of Components vs. Log Likelihood

From Figure 49 we can see that 11 components yields the highest Adjusted Random Score; however, at 9 components the score isn't that different either. We almost get a Random Score of 0.6 at 11 Components which shows that maybe for this dataset EM is better performing than K-Means. I've graphed the clustering of EM with 11 clusters in Figure 51.

From Figure 51 we can see that the points closest to y=0 are clustered better than they were with K-Means. With 11 components we are able to achieve a Random Score of almost .68 again reinforcing the idea that EM is preforming better than K-Means.

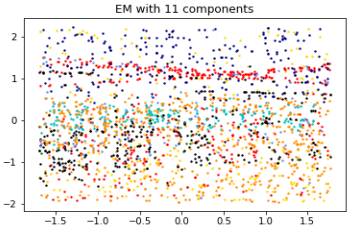

Figure 51: EM w/ 11 Components

Image-Segmentation (Principle Component Analysis)

The next algorithm to run is PCA and like last time I looked at how many components it took to cover most of the variance in the data. With the first component 42.34% of the variance is covered, with the first 2 components 58.54% of the variance is cover and with the first 3, 68.50% of the variance is covered. At 9 features 94.77% of the variance is covered and after that point the variance covered increases naturally but doesn't add anything substantial, so we will not include them going forward. Now lets take a look at what happens to K-Means' Adjusted Random Score as the number of PCA components is varied. You can see what happens in Figure 52.

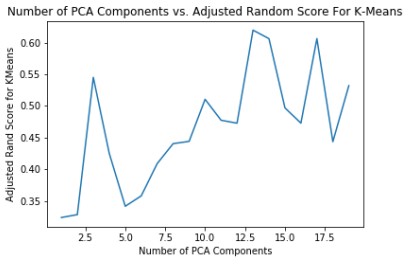

Figure 52: Number of PCA Components vs. Adjusted Rand Score for K-Means

From Figure 52 we can see that the highest Random Score was received at 13 components. With 13 components we are able to get about 98% of the starting features space, past 13 components we start seeing the effects of the curse of dimensionality. Another observation we can make is that when we applied PCA and then ran K-Means we are able to achieve a higher Random Score of over 0.60. One more final observation that we can see is that at 2 components less than 0.35 is out score but at 3 the score shoots up to about 0.55, the Figures 53, and 54 should show why that is the case.

Figure 53 shows us what the clustering looking like for K-Means with only 2 PCA Components. We can is that the points are all either jumbled together or scattered far apart (not good). In Figure 54 we can see the clustering for 3 PCA components and the clustering looks clearer thus why it yields a higher score.

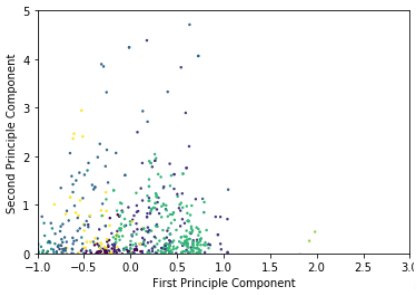

Figure 53: K-Means Clustering for 2 PCA Components

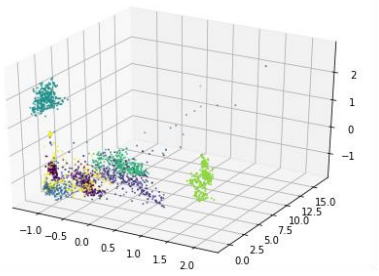

Figure 54: K-Means Clustering for 3 PCA Components

Now moving away from K-Means lets take a look at EM. Figure 55 show the impacts on Random Score when the number of PCA components is varied. From the figure we can see that the max Rand Score is achieved at 13 components, making EM consistent with K-Means. Now lets take a look at what the clustering looks like for K-Means with 8 clusters and EM with 11 clusters. The graphs of the visualizations are available in Figures 56 and 57. Looking at the figures you can see that the clusters look to be easier to separate, more so for K-Means as the clusters seem to be more defined.

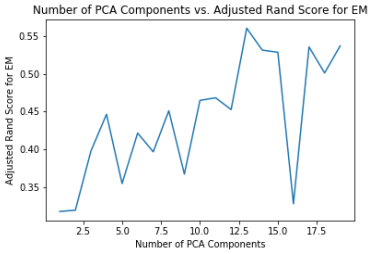

Figure 55: Number of PCA Components vs. ARS for EM

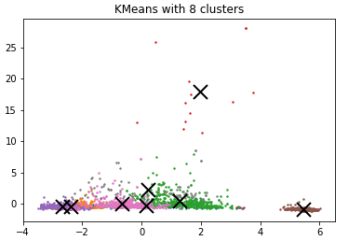

Figure 56: K-Means with 8 Clusters and 13 PCA Components

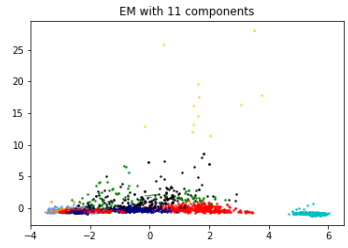

Figure 57: EM w/ 11 Components and 13 PCA Components

To conclude PCA, 13 PCA components yields greater improvements for K-Means (greater than 0.60) than it does for EM (looks to stay around 0.60 where it was without PCA).

Image-Segmentation (ICA)

Now let's take a look at ICA, again I will be using Sci-Kit Learns FastICA implementation. Like with PCA let take a look at how K-Means behaves with ICA. The impact on Rand Score when ICA components is varied is shown in Figure 58. From the figure we can see that 5 components yields the greatest Random Score of about 0.55 not really an improvement from our base K-Means. I'm kind of interested in see why the score goes from around 0.30 at 2 components to below 0.25 at 3 components. The Figure for 2 and 3 components is shown in Figures 59 and 60. From the figure its very evident why 2ICA Components yielded a higher Score than 3, with 2 components we see how clean the clusters are formed and with 3 components we see that there is bunch of points from various clusters intertwined together making it difficult to separate the clusters. Now let's take a look at EM in Figure 61. From the graph again 5 components seem to yield the highest ARS for EM as well. Let take a look at the clustering K-Means at 8 Clusters (Figure 62) and EM with 11 components (Figure 63) both with 5 ICA components. Looking at the figures we can see why there isn't much improvement in Random Score when we use ICA to do dimension reduction, the clusters look fine but the look similar to what they looked like without ICA.

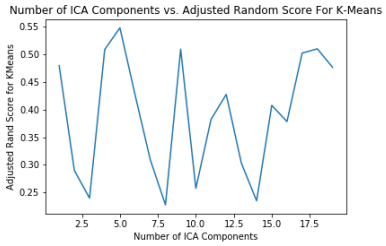

Figure 58: Number of ICA components vs. ARS K-Means

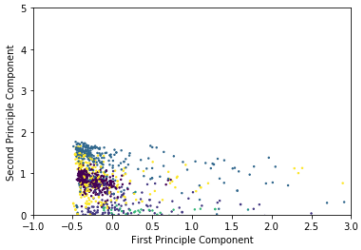

Figure 59: K-Means with 2 ICA components

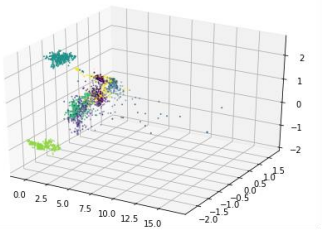

Figure 60: K-Means with 3 ICA Components

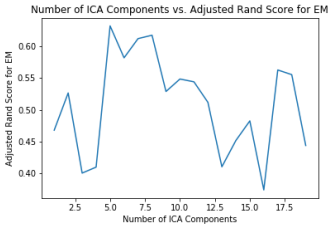

Figure 61: EM Number of ICA components vs. ARS

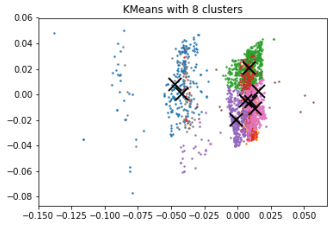

Figure 62: K-Means w/ 8 Clusters and 5 ICA Components

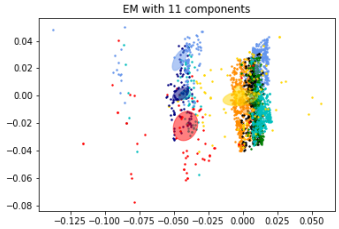

Figure 63: EM w/ 11 Components and 5 ICA Components

Image-Segmentation (Random Projection)

Now let's take a look at Random Project (RP). Let's first take a look at how K-Means' and EM's Random Score changes as the number of RCA components, you can see the relationships with different runs in Figures 64 and 65 respectively.

From Figures 64 and 65 we can see there is a lot of variance from run to run, we can also that one average as the number of ICA components increases the Random Score also increases for both graphs/ Clustering Algorithms. We can deduce from our previous observation that unlike with the Voice-Classification Dataset we need as many components as we have features. Another observation that we can make is that for both graphs the max scores never go past that which was achieved with PCA, and we can reasonably understand why. The whole point of RP is that it gives up Accuracy to increase Speed. Unlike with the previous dataset I chose not to show the improvements that tweaking density yields as increasing density is more expensive when it comes to speed and thus negates the benefits of RP.

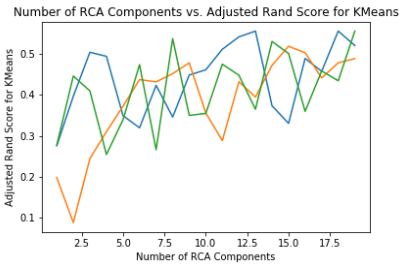

Figure 64: Number of RCA Components vs. ARS of K-Means

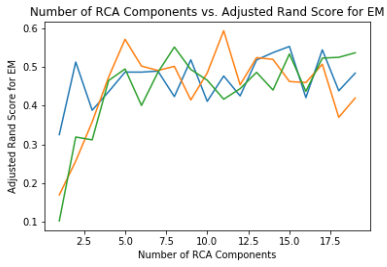

Figure 65: Number of RCA Components vs. ARS of EM

Image-Segmentation (Autoencoders)

The last thing I will look at for this dataset is Autoencoders. The first thing I want to take a look at is how the number of features generated by the Auto Encoder impacts the Random Score for both K-Means (Figure 66) and EM (Figure 67).

From Figures 66 and 67 similar to what we saw in the Voice-Classification dataset we can see that number of features that are need to reach the max score that can be reached by Autoencoder on both clustering algorithms increased past the base number of features and we didn't even see that much of an improvement in the random score.

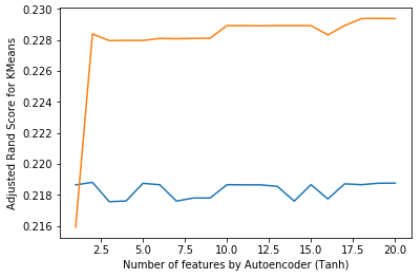

Figure 66: Number of Features Generated by Autoencoder vs. ARS K-Means

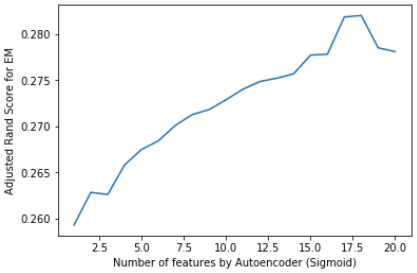

Figure 67: Number of Features Generated by Autoencoder vs. ARS EM

The next thing I wanted to look at was how the clusters looked like for both K-Means and EM when I add a sparsity constraint (which allows most of nodes in the neural net will stay inactive) to the Autoencoder. Figure 68 shows K-Means' clustering and Figure 69 shows EM's clustering. Now that interesting! The clusters are very defined, you can however see that for EM some of the clusters area broken by other clusters. The major downside of AE is the time it takes to run. Because that and that fact that the ARS was not able to break 0.30 AE is the worst performer of the Dimension Reduction algorithms (But it looks so clean.)

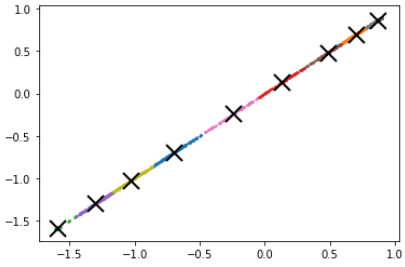

Figure 68: AE with Sparsity Constraint K-Means

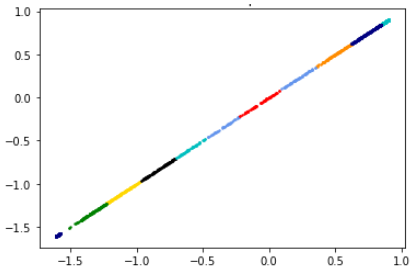

Figure 69: AE with Sparsity Constraint EM

Conclusion

For the Voice-Classification dataset here's what we saw:

- K-Means and EM both preformed about the same
- PCA was the best dimension reduction algorithm
- ICA performed slightly better than RP
- Autoencoder was the worst preforming DRA
- And when we looked at the Neural Net PCA again performed the best.

For the Image-Segmentation dataset here's what we saw:

- K-Means was a better performer than EM sometimes; however, on average, they were about the same
- PCA was again the best DRA
- RP was second best performing
- ICA after RP
- Autoencoder was worst again