ANALYSIS OF CLUSTERING AND DIMENSIONALITY REDUCTION ALGORITHMS

NISHANT ROY

CS 4641

Datasets Chosen

OptDigits: This dataset is comprised of 8x8 pixel matrices generated from normalized bitmaps of handwritten digits, where the goal is to predict the digit (0-9) represented. The data is complete. There are 64 attributes (excluding class) ranging from 0-16 each, 10 possible values for the class (0-9) and 5620 total instances. The high dimensionality makes clustering a challenging task, since distance measurements become less precise with a large number of dimensions. Therefore, dimensionality reduction algorithms are likely to help improve the performance of clustering algorithms on this dataset.

Training computers to identify handwritten figures is an interesting problem within the field of computer vision, because of the tremendous variations in handwriting, as well as how similar some digits look (8 looks much like a 6 or 9, 4 and 9 look similar, etc.), leading to noise in the data. In the absence of labels, clustering algorithms are an effective way to identify similar looking digits.

Wine Quality: This dataset is comprised of different attributes of wine (acidity, sugar content, density, alcohol content, etc.), where the goal is to predict the quality (0-10) of the wine. The data is complete, however, the only values for the quality are from 3 to 9, so, there are really 7 possible values for the class. There are 11 continuous, numeric attributes (excluding class), and 4898 total instances. The dimensionality is lower than OptDigits, so reduction might not be as important. However, since certain attributes may be correlated (e.g.: free sulfur dioxide and sulfur dioxide), feature selection may still improve performance of clustering algorithms.

Clustering Algorithms

K-Means

The Simple K-Means algorithm from Weka was used for both datasets. K-Means picks k centroids, and assigns the points to the nearest centroid based on some distance metric (Euclidean distance for my experiments). Since clustering is an algorithm for unsupervised learning, i.e., learning data without labels, the class attribute was ignored. Weka's classes to cluster evaluation method was used to measure how well the algorithm clustered the same classes together. This evaluation method looks at the majority class in each cluster and assigns that to be the label for that cluster, then calculates the accuracy by seeing how many instances match up with the label of their cluster.

Additionally, the within-cluster sum of squared errors is also tracked and the elbow method is used to identify optimal values for k. The graph of the SSE often looks like an arm, and the "elbow" in the arm is chosen as the best value for k. This value represents the point at which we start to see diminishing returns. SSE will continue to fall as k increases – it approaches 0 as the value of k approaches the number of instances, since if each instance is its own cluster, then the distance to the centroid is always 0.

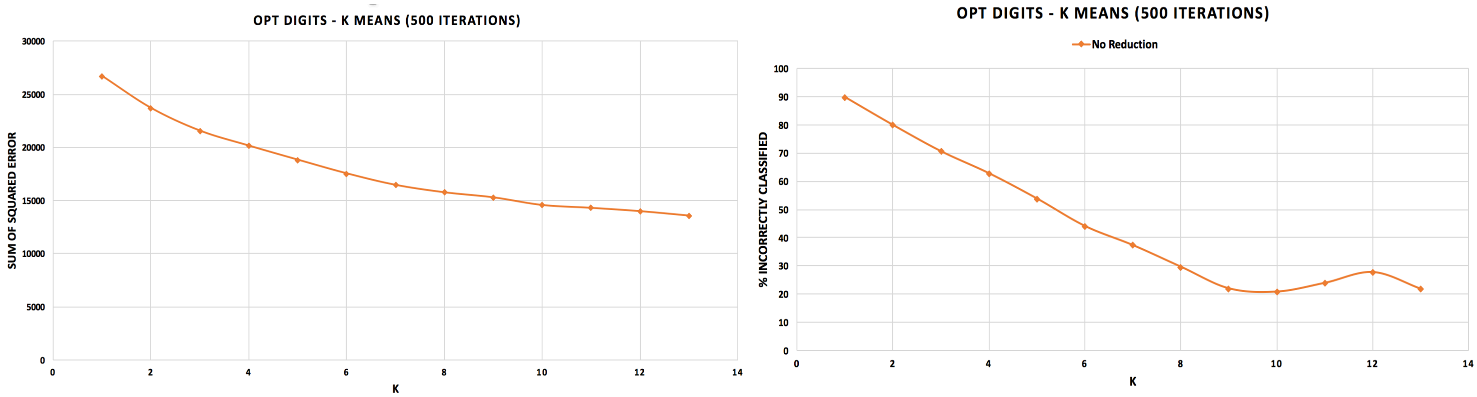

Figure 1: K-Means OptDigits [LEFT]: SSE; [RIGHT]: Error

Looking at Figure 1 [LEFT], we see that the SSE decreases quite consistently, and there is no clear elbow. This makes it hard to choose an optimal value for k. However, when we look at Figure 1 [RIGHT], we see that the lowest value for error is reached when k = 10; the error starts to rise after. As a result, 10 is the optimal value for k in this dataset. In this case, we have some knowledge about the dataset, i.e., we know there are 10 possible classes, so this value for k seems like a good choice, since it gives us enough clusters for all our classes.

However, this is not necessarily the case. We might have multiple clusters that point to the same class, and, we may have cases were similar looking numbers are inaccurately clustered together. For example, the algorithm might put a slanted 4 and a slanted 9 together, and a straight 4 and straight 9 together, since the pixel arrangement might be closer in those cases. This is supported by the fact that the error dips once again at k = 13. With 13 clusters, we may actually get at least one cluster for each class rather than multiple clusters for one class and none for others, which helps address the error described here. The high dimensionality might also impact our clustering, since we may have attributes that are not very relevant, but we are considering them anyway.

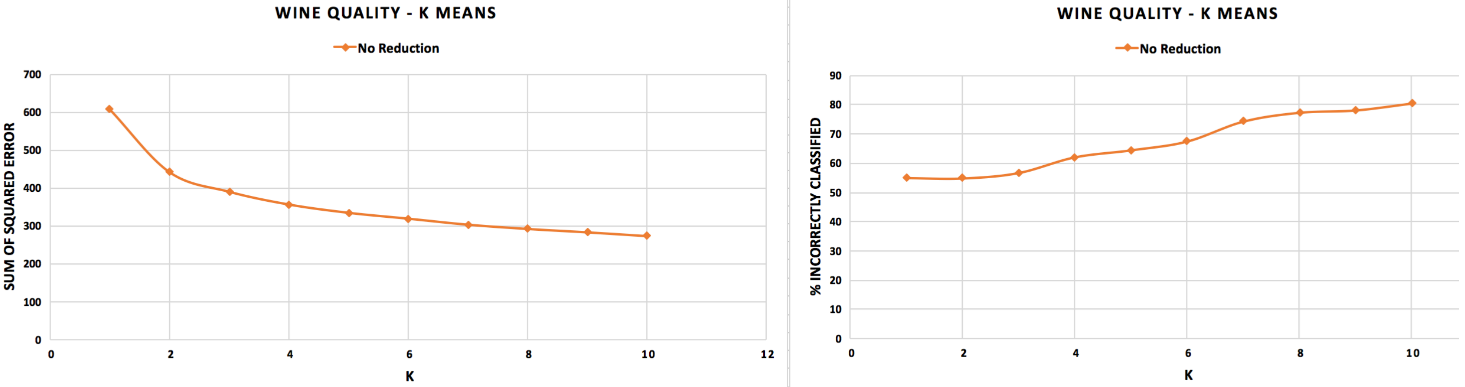

Figure 2: K-Means Wine Quality [LEFT]: SSE; [RIGHT]: Error

Looking at Figure 2 [LEFT], we see a clear elbow at k = 2, indicating this is the optimal k-value for this dataset. As k increases, the error continues to rise. This is likely due to the fact that as we increase k, the clusters become more homogenous, i.e., the distribution of instances in each cluster is similar, compared to fewer clusters, which makes differences in attributes more visible. Even though there are 7 possible values for the class (from our knowledge about the dataset), because attributes of this dataset may be correlated, fewer clusters perform better. The error is very high (over 50% at best), indicating that the clustering did not separate the classes very well. This is likely due to the correlation between attributes in the dataset, which makes it harder to separate instances.

For both datasets, increasing k eventually leads to increased error, which is expected due to the curse of dimensionality. As we increase k, we are considering higher dimensions, which means we need a lot more data to train our model well.

Expectation Maximization

The Expectation Maximization (EM) algorithm calculates the likelihood of each instance being in a particular cluster, and then assigns the instance to the cluster with highest probability. This allows sharing of instances between clusters, which is beneficial because in some cases, it may appear that a cluster could belong to more than one cluster.

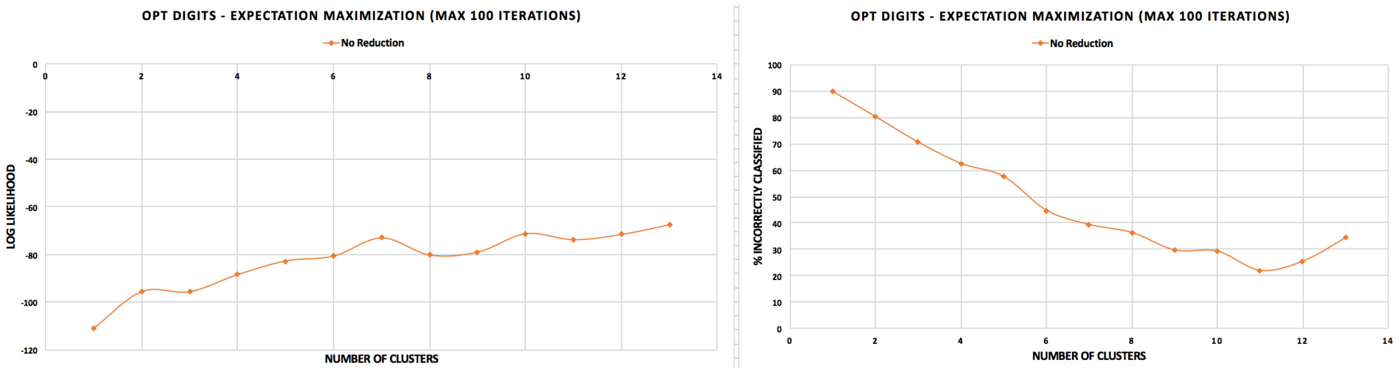

Figure 3: EM OptDigits [LEFT]: Log Likelihood; [RIGHT]: Error

Looking at Figure 3 [LEFT], we see 3 local maxima at 2, 7, and 10 clusters. It would be interesting to see if the likelihood continues to increase after 13 clusters, because we may find that we increase the likelihood function. Clearly we are not very confident in our clustering here, with a maximum log likelihood value of -71.32 for 13 clusters. Looking at Figure 3 [RIGHT], we see that EM produces the lowest error with 11 clusters, compared to 10 for K-Means. K-Means is a little more accurate (20.78% error) than EM (21.99% error), and it is also significantly faster (1.22 seconds vs. 20.1 seconds). I expected EM to be more accurate, since, as described earlier, certain handwritten digits may be very close to two clusters, and by assigning probabilities to them, we might get more accurate clustering. At 13 clusters, which was found to have the highest likelihood, the error rises to 34.36%, which is notably higher than the error for K-Means. One explanation for this may be the high dimensionality – with 65 attributes, there may be some that are not very relevant, and considering them for our clustering could affect our accuracy.

Further experimentation with a higher number of clusters might be helpful in maximizing the likelihood function; it is possible that the clusters we create fall prey to similar looking numbers, and by creating more clusters, our classes may be better separated.

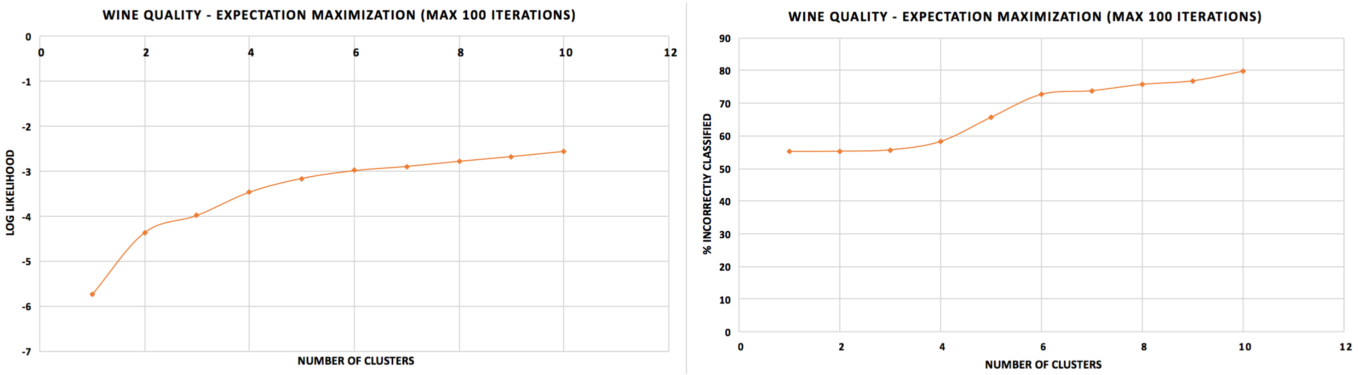

Figure 4: EM Wine Quality [LEFT]: Log Likelihood; [RIGHT]: Error

Looking at Figure 4 [LEFT], we see that the likelihood function value is always increasing, however, between 4 to 6 clusters, the rate of increase falls considerably. We hit the point of diminishing returns at roughly 5 clusters. In this dataset, we see that we have a much higher value for likelihood than OptDigits (at 10 clusters, our log likelihood is -2.5), indicating we are much more confident in our clustering. This is likely due to much lower dimensionality (11 attributes vs. 64) allowing us to more accurately calculate the Euclidean distance, allowing us to more confidently cluster the instances. Looking at Figure 4 [RIGHT], we see that the error is constantly increasing and is very high (over 50% at best), much like K-Means. The error is marginally lower than the error in K-Means, but they are both inaccurate enough to not be trustworthy models. Once again, K-Means was a lot faster (~10 times as fast as EM). EM was unable to separate the classes very well, probably because the correlation between attributes made it harder to separate different quality wines.

Both EM and K-Means should perform better if we can remove attributes that are not very relevant to the clustering problem, so, the performance of both algorithms should increase once dimensionality reduction algorithms are applied.

Dimensionality Reduction/Feature Transformation Algorithms

Feature transformation and feature selection algorithms are used to identify features that are relevant to the dataset and focus on them in order to reduce the dimensionality of the dataset, thus making clustering an easier task. Four different algorithms were used for this purpose: Principal Component Analysis, Randomized Projection, Independent Component Analysis, and Information Gain Attribute Evaluation.

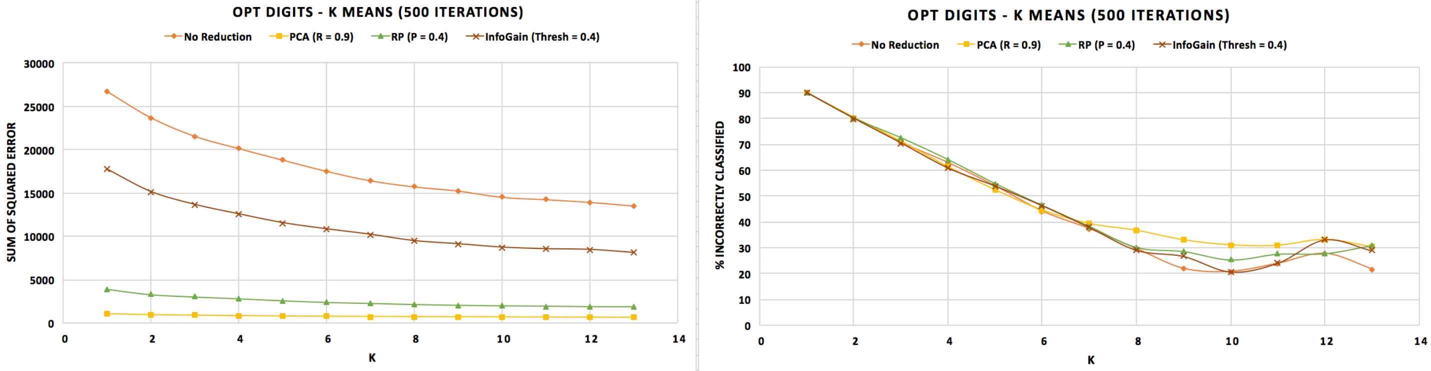

Figure 5: K-Means OptDigits [LEFT]: SSE; [RIGHT]: Error

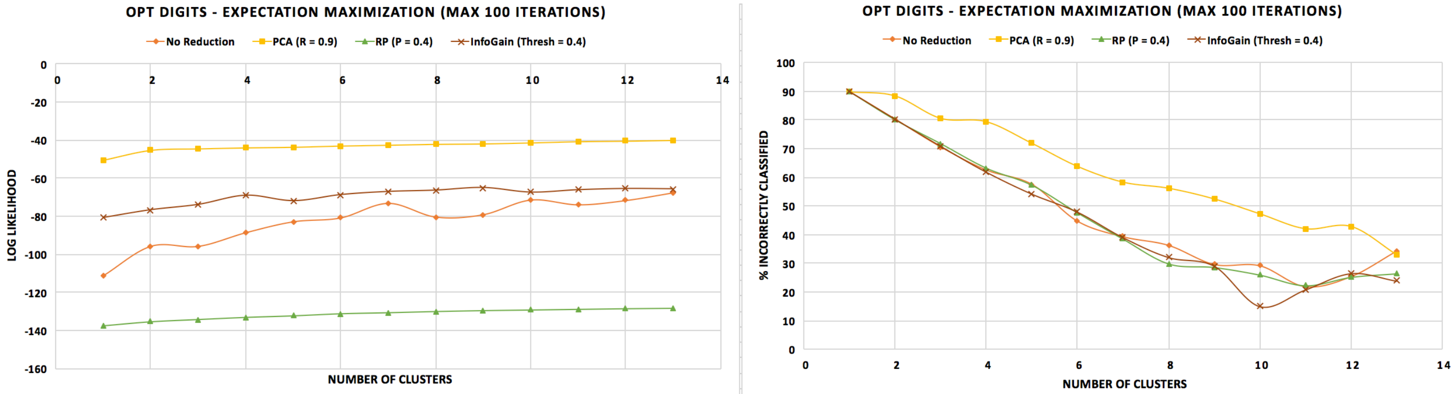

Figure 6: EM OptDigits [LEFT]: Log Likelihood; [RIGHT]: Error

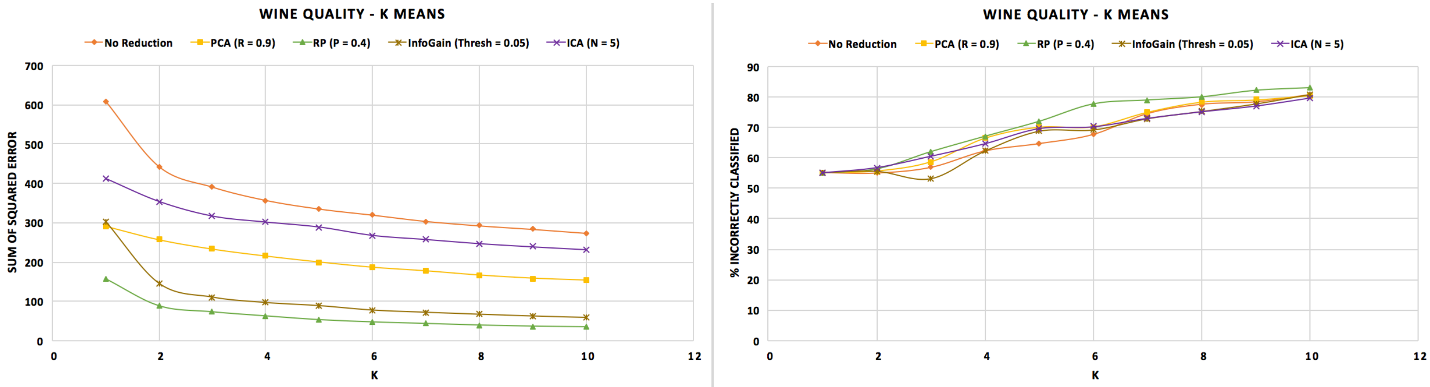

Figure 7: K-Means Wine Quality [LEFT]: SSE; [RIGHT]: Error

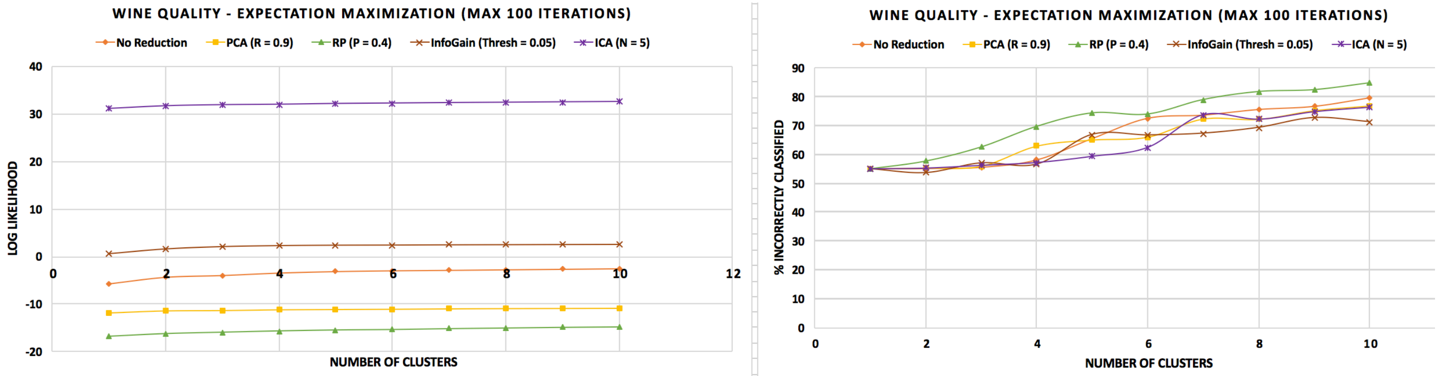

Figure 8: EM Wine Quality [LEFT]: Log Likelihood; [RIGHT]: Error

Principal Component Analysis (PCA)

The PCA algorithm projects our data into a subspace with lower dimension by choosing the dimensions along which variance is maximized, hence choosing the linear combinations of attributes that are the most relevant to our dataset. It does so by generating the covariance matrix and computing the eigenvectors in descending order of eigenvalues. The principal eigenvector is the one with the largest eigenvalue, and represents the most relevant attribute. To reduce it to a n-dimensional space, we consider the first n eigenvectors, which represents the most relevant linear combinations of attributes. While running PCA, the variance retained was constant at 90%, and the number of attributes per linear combination was not limited.

Looking at Figure 5 [LEFT], we see that PCA reduces the SSE for OptDigits to almost 0 from nearly 30000 without reduction. The low SSE shows that the clusters were much more concentrated around their respective centroids, so the clustering was actually improved. PCA reduced the number of attributes to 33. The eigenvalues were skewed towards the left, with the top three being 7.2, 6.3, 4.83, and most of the remaining ranging from 0.44 to 1.7. This indicates that certain combinations of pixels were extremely relevant, while others were less important. The error from classes to cluster evaluation increased, however, it decreased for k = 13, so perhaps testing for higher values of k might have shown improved accuracy in classifying OptDigits.

In Figure 6 [LEFT], we see that PCA increased the likelihood value for OptDigits significantly, so it improved the performance for EM as well. Figure 6 [RIGHT] shows that the error, much like K-Means, increased after reduction, but dipped very sharply for k = 13, which further supports the hypothesis that more than 13 clusters might improve accuracy for OptDigits.

In Figure 7 [LEFT], we see that PCA lowers the SSE for Wine Quality as well, albeit not as drastically as it did for OptDigits. Unlike OptDigits, most of Wine Quality's attributes are actually relevant, they are just correlated, which means that combinations of the attributes may be better suited for clustering, removing attributes is not so important. We retained 8 attributes for this dataset. From Figure 7 [RIGHT], we observe that the error in clustering is very high once again, so while we improved the concentration of our clusters, we did not improve our accuracy. Once again, the eigenvalues were skewed to the left – the top eigenvalue was 3.22, while the rest ranged from 0.6 to 1.6. This means that one particular combination of attributes was much more important than others.

Looking at Figure 8 [LEFT], we see that PCA actually reduces the likelihood of our Wine Quality clustering, so it decreased EM performance for this dataset. The error was slightly lower than it was before reduction, but still high enough (>50% at best) to make the model untrustworthy.

For OptDigits, K-Means was faster after PCA, while EM was unexpectedly a bit slower. Clustering with 33 attributes rather than 65 should have been much faster, however, EM did not exhibit this behavior, possibly because the linear combinations were complex and calculating the likelihood value took more time. Experimenting with different variance retention values as well as increasing the value for k/number of clusters may help improve evaluation accuracy.

For Wine Quality, both algorithms ran faster after PCA, which is the expected behavior, since there are now fewer attributes to consider. The accuracy did not vary much for either algorithm, and was extremely low. Trying different values for variance retention may yield improved performance by restricting or allowing attributes with lower eigenvalues.

Randomized Projection (RP)

The RP algorithm projects our data from d to n dimensions by using a n x d dimensional matrix. The matrix is randomly generated from a given distribution and some seed value. For my experiments, Weka's Sparse1 distribution was used and 40% of the attributes were retained. The seed value was varied between {42, 25, 10} and the average of the results was taken. RP maintains the distance between pairs of instances well. It lowered the dimensionality for OptDigits from 65 to 25 attributes, and for Wine Quality from 11 to 4 attributes.

From Figure 5, we see that RP lowered the SSE for OptDigits notably as well, almost as much as PCA, with less attributes, making the clustering process faster. The error is higher than the non-reduced data and appears to increase after k = 12 whereas the other algorithms all decrease, so RP worked the best with k = 10, which we know to be the exact number of classes. It is possible that RP chose the attributes in such a way that the 10 clusters each represented a different class, but unlikely since the error is higher than non-reduced data. From Figure 6, we see RP negatively impacted EM for OptDigits, lowering the log likelihood significantly to about -130. However, the error was lower than when no reduction was performed.

From Figure 7, we see that RP lowered the SSE for Wine Quality the most, creating the most concentrated clusters. However, RP also produced the highest error, almost 85%, so while it helped separate clusters better, it did not improve the evaluation. In Figure 8, we see that RP gave the worst log likelihood for Wine Quality, and clearly the highest error. RP is definitely not well suited for this dataset.

RP works well for OptDigits since it reduces the dimensionality, which helps hone in on relevant attributes. However, for Wine Quality, we already have low dimensionality, and we want to find the attributes that are correlated and have a linear combination of them, which RP is unable to effectively do. Experimenting with higher variance retention might improve the performance of RP, since more information will be retained. This, however, would not lower clustering time as much, since more attributes would need to be considered.

Information Gain Attribute Evaluation (IG)

The IG algorithm ranks the dataset's attributes based on the information gained from each one, and then selects the top n based on some threshold to create an n-dimensionality subspace. I used a threshold value of 0.05 for Wine Quality, and 0.4 for OptDigits. Wine Quality was reduced to 8 attributes, and OptDigits was reduced to 26. The Information Gain values for OptDigits were evenly distributed, with all 26 ranging from 0.41 to 0.67. Similarly, for Wine Quality, all 8 attributes lay between 0.06 and 0.2.

From Figure 5 and 7, we see that IG lowered the SSE for both datasets a little bit. For OptDigits, the error was a little higher in some places, but roughly the same overall. For Wine Quality, the error was a little higher overall. IG helped produce more concentrated clusters, but the clusters did not perform well under evaluation, so it was unable to accurately separate different labels into different clusters.

From Figure 6 and 8, we see that IG lowered the log likelihood for both datasets, reducing the confidence in our clustering by impacting the evaluation of the distance metric. However, the error for both datasets was lower than before reduction. In fact, for both datasets, 10 clusters provided the lowest error of all (~15% for OptDigits, ~71% for Wine Quality). By picking the attributes that give us the most information about our data, we are able to cluster more effectively, which reduces the error as compared to other algorithms which reduce dimensionality by other metrics.

Once again, both clustering algorithms performed very poorly under evaluation for the Wine Quality dataset, but were fairly accurate for OptDigits. OptDigits has a lot more attributes (65 vs. 11), which probably makes it easier to make distinctions. However, Wine Quality has a much higher ratio of data to dimensionality, so I would have expected the clustering to perform better on Wine Quality, because it should've been less impacted by the curse of dimensionality.

Independent Component Analysis (ICA)

ICA is a technique used for the separation of linearly mixed sources. ICA attempts to maximize the independence of the estimated sources in order to find them. This can be done either through minimizing mutual information or maximizing non-Gaussianity. Weka uses FastICA, which maximizes non-Gaussianity by rotating whitened data (linear transformation where the elements are all uncorrelated and have variance 1). The number of attributes retained was set to 5 in order to get the most relevant attributes and ignore all others.

Unfortunately, the FastICA algorithm on Weka failed when I attempted to run it with the OptDigits dataset, and the FastICA implantation on MATLAB did not generate reduced data, so I was unable to test the clustering algorithms with the ICA reduction.

From Figure 7 [LEFT], we can see that ICA lowered the SSE for the Wine Quality dataset almost as effectively as RP. From Figure 7 [RIGHT], we see that the error was roughly the same or lower than before reduction was applied. By choosing only the top 5 attributes, we reduced dimensionality substantially which made clustering easier, and we focused on the most relevant attributes, which helped separate labels better, thus producing the lowest error values.

From Figure 8 [LEFT], we can see that ICA produced the highest log likelihood by a very large margin (~30 better than the next). This means that the clusters produced by EM after ICA had extremely high probability, and we were very confident that we had clustered well. The error is still very high as we can see in Figure 8 [RIGHT], however, for lower values of k ($k \leq 6$), ICA has the lowest error, spiking after.

Once again, the clustering algorithms did not provide a satisfactory accuracy for the Wine Quality dataset. The feature selection was unable to effectively find the best subset of attributes required to cluster Wine Quality in a manner that performed well under the classes to cluster evaluation method.

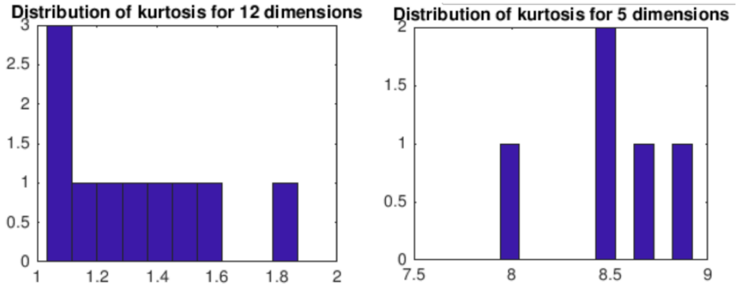

Figure 9: Kurtosis before/after ICA for Wine Quality

Before ICA, there is one attribute that has a value of 3 for Kurtosis, indicating that it is normally distributed and mesokurtic. The others are all 1 (platykurtic with an excess kurtosis of -2), so there are likely to be fewer and less extreme outliers for these attributes.

After ICA, the highest kurtosis is 2 (platykurtic with an excess kurtosis of -1), and the rest are all 1 (platykurtic with an excess kurtosis of -2), which means there are no normally distributed attributes anymore, but there are still likely to be fewer and less extreme outliers for the Wine Quality dataset.

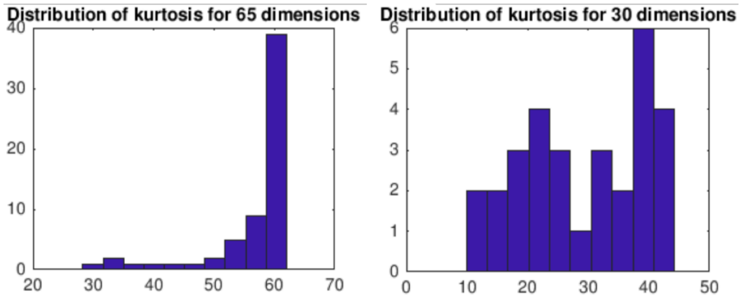

Figure 10: Kurtosis before/after ICA for OptDigits

Before ICA, several attributes have a value for kurtosis greater than 3 (leptokurtic with high excess kurtosis), one almost as high as 40, which means a high likelihood of a higher number of and more extreme outliers.

After ICA, there are only 3 attributes with a value greater than 3, with the highest being 6 (leptokurtic with excess kurtosis of 3), 3 attributes with kurtosis exactly 3 (mesokurtic), and the rest being less than 3 (platykurtic with negative excess kurtosis). So, ICA lowered the likelihood of extreme outliers in the OptDigits dataset.

Neural Network Learner

In order to analyze how clustering and dimensionality reduction can impact the performance of the Neural Network classifier, I used Weka's Multilayer Perceptron Classifier on the OptDigits dataset after reduction and clustering. For comparison's sake, I used the results and the tuned hyper-parameters from the analysis of supervised learning algorithms earlier in the semester. Learning rate = 0.4, momentum = 0.5, and 1 hidden layer with 5 nodes trained over 800 epochs. A 70/30 split was used to train and test the neural network classifier, and the error reported is the test error, while the time reported is the training time. For the clustering algorithms, a value of 10 for k and number of clusters was used, because that gave the best results from earlier experiments.

No Dimensionality Reduction	Error %			Time		
	7.0581			30.77		
	No Clustering		K-Means		EM	
	Error %	Time	Error %	Time	Error %	Time
PCA	8.4223	22.34	0.4152	26.42	3.9739	26.16
RP	12.159	19.74	3.5587	23.92	5.4567	23.32
IG	9.6085	19.33	2.4318	25.68	7.7699	22.99

Figure 11: Neural Network performance with reduction and clustering

Unfortunately, the FastICA algorithm on Weka failed when I attempted to run it with the OptDigits dataset, and the FastICA implantation on MATLAB did not generate reduced data, so I was unable to test the neural network learner with ICA reduction.

When run with only reduction, the error increased, so simply reducing the dimensions does not improve the neural network's performance. However, the training time went down roughly 33% after the feature transformation was applied. This is a tradeoff that may be worth considering – if dimensionality reduction does not cause a significant increase in accuracy, and it is capable of reducing the time taken to train the network considerably, it may be helpful to perform the reduction. This, of course, depends on individual needs.

The dimensionality reduction algorithms work by selecting either the most relevant attributes or the most relevant linear combinations of attributes, and discarding the less important ones. I expected the accuracy of the neural network to improve with this step, because lower dimensionality reduces the size of the network, and hence reduces the amount of data required to train it. Furthermore, by considering fewer attributes, the network is less likely to overfit, allowing for better generalization and, hence, improved accuracy is expected. However, reduction algorithms may remove information that is important to make the distinction between classes if this information is present in attributes with low variance, low information gain, etc., thereby impacting the accuracy of the neural network. Reduction may be useful for some datasets, but for this particular dataset, it is not effective. To an extent, in fact, the weights of the network may perform linear transformation like certain feature transformation algorithms (like PCA), so reduction isn't necessarily a crucial step for neural network learning.

When clustering was used in tandem with dimensionality reduction, the performance was significantly better, except for the combination of IG and EM clustering, where the error was slightly higher than without any reduction or clustering. In almost every other situation, however, the performance was notably better, with the combination of PCA and K-Means clustering giving an almost perfect model, with only 0.42% error. While the training time was slightly higher than just dimensionality reduction, it was still lower than the training time without reduction or clustering, so, clearly, clustering along with dimensionality reduction is extremely beneficial to neural network learning.

Clustering can give some semblance of organization to otherwise very complex data. Neural networks are very powerful classifiers capable of learning and simulating very complicated, non-linear functions. By adding some level of organization through the clustering, the neural network is capable of modeling more complex functions faster. The dimensionality reduction helps in two ways. As we saw earlier, these algorithms can improve the performance of clustering algorithms, lowering the sum of square error and increasing the likelihood function value, which allows stronger, more concentrated clustering, and higher confidence levels in our clusters, because in lower dimensions, it is easier to calculate the distance metric. It also helps make the neural network simpler, which reduces overfitting as well as training time. Clearly, the combination of these two steps is likely to improve the performance of the neural network learner, as seen with the OptDigits dataset here.

Conclusion

The high dimensionality (65 attributes) of OptDigits affected the performance of the clustering algorithms, with K-Means having an extremely high SSE, and EM having a very low log likelihood. Both of these values indicate that the clusters were not concentrated and well differentiated. Comparatively, K-means had a much lower SSE and EM had a much higher log likelihood for Wine Quality, which had a notably lower dimensionality (12 attributes). However, the classes to cluster evaluation yielded unexpected results. High dimensionality makes the distance metric harder to precisely evaluate, making it harder to accurately cluster instances, which should result in higher error. The high dimensional OptDigits had low error (< 30 %), while the low dimensional Wine Quality dataset had very high error (> 50% at best). Wine Quality has certain attributes that may be correlated, which could cause overlapping clusters. The high number of attributes in OptDigits may be providing sufficient information to separate the instances into the required labels.

Reducing dimensionality causes SSE to fall drastically for OptDigits, and almost always increases the log likelihood (RP reduced it significantly). Both of these values indicate that the clusters are more well defined than before, and more concentrated around the centroids. SSE fell for Wine Quality as well, however, log likelihood decreased for PCA and RP, increased a bit for IG, and dramatically for ICA – likely because ICA is very effective at separating individual sources of data. The evaluation error showed the same behavior – low for OptDigits, very high for Wine Quality. The errors increased slightly for K-Means with OptDigits and stayed roughly constant with Wine Quality. When we can only pick one cluster for each instance, and we are reducing the attributes available for clustering, we have less information to accurately choose a cluster. For EM, the errors decreased for OptDigits (except for PCA), and decreased for K-Means (except for RP). Since clusters can share instances, and we have lower dimensionality which gives us increased accuracy in evaluating the distance metric, EM is able to define clusters for each instance with higher accuracy.

By applying dimensionality reduction algorithms, we do not see an improvement in the performance of our neural network classifier, in fact, we see a slight dip in accuracy, and a considerable dip in training time. This may be a tradeoff worth considering if we value speed over accuracy, and the accuracy dip is not too large. When we add the results of clustering after dimensionality reduction as a feature, our neural network performs remarkably better than before – faster and with higher accuracy. The clustering adds some structure to our otherwise unstructured data, and the lower dimensionality reduces the size of the neural network, allowing for faster computations, lower likelihood of overfitting, better generalization, and, overall, much better performance.

This analysis showed that clustering and dimensionality reduction algorithms may perform differently based on the nature of the attributes of the dataset and its dimensionality. Lower dimensionality almost always guaranteed faster computation, but not necessarily more accurate results. It also showed that preprocessing and clustering the dataset can be very helpful in training a neural network, and getting faster, more accurate results.