CS 4641 - Assignment 3

By: Sean Chua (schua8)

1 Introduction and Overview

1.1 Datasets

For this paper, we will be making use of two datasets that we've used before: one in the unsupervised learning paper and both in the supervised learning paper. In particular, the datasets we will be looking at are:

Breast Cancer Dataset - This is a dataset of 569 instances that describe characteristics of breast cancer data, provided by researchers at the University of Wisconsin. This dataset contains 32 attributes, which include ID, diagnosis, and ten features that were calculated for each breast cancer cell nucleus, such as radius, texture, concavity, symmetry, and more. Along with these 32 features, there are 2 class types of "Benign" and "Malignant", which signify the doctor's diagnosis. I chose this dataset because creating a machine learning algorithm that could diagnose breast cancer as well or even better than a doctor could would have vast implications for fighting breast cancer. This is because using a machine learning algorithm, as opposed to hiring a doctor to do a diagnosis, is much less financially expensive and could lower the barrier for people to check on their health. This could help diagnose early forms of breast cancer, save patients' lives, and save them a lot of financial and physical problems. Lastly, I wanted to work on a significantly smaller dataset than the other one that I chose in order to see how training size and complexity can affect learners.

Handwritten Digits Dataset - This dataset has 5620 instances of handwritten digits (ranging from 0 to 9) that were extracted by researchers from the Bogazici University using preprocessing programs to yield normalized bitmaps of handwritten digits from a preprinted form. A total of 43 people contributed their writing to this dataset, 30 of which were included in the training set and 13 of which to the test set. These 32x32 bitmaps were divided into 4x4 blocks (which don't overlap), and the number of "on" pixels are counted in each block. This yields an 8x8 input matrix in which each element is an integer in range [0,16]. As a result, there are only 64 input attributes and one class attributes, which is a great reduction of dimensionality. This dataset is also much more complex and larger than the breast cancer dataset, so it will be interesting to see the effect on testing error as a function of training size for the numerous algorithms. I found this dataset particularly interesting because it represents something that humans find intuitive but is much more difficult for a computer to classify. Image classification has also become one of the most interesting applications of machine learning nowadays, as technologists have been able to implement it in facial recognition to unlock phones, handwriting recognition, and more. As I was looking through the description of the handwritten digits dataset, I found it intriguing to see how much clever preprocessing it took to create such a dataset so that a machine could learn how to recognize handwritten digits. Handwritten digits are somewhat standardized across cultures, but I would love to explore this space further with an analysis on other forms of handwriting (not just within English).

Note: all datasets were split into 70/30 training and testing sets.

2 Clustering

2.1 Introduction to Clustering

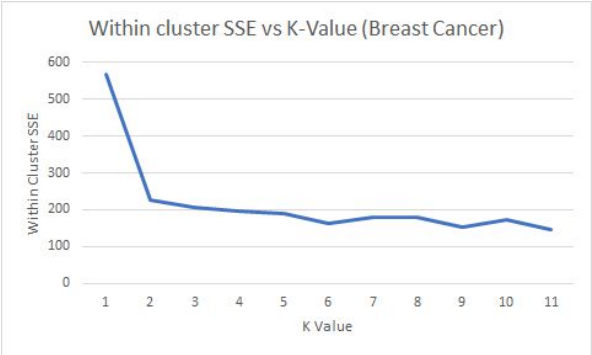
In the context of unsupervised learning, clustering refers to the general task of taking a set of data consisting of points or objects, and grouping the objects into clusters. For each cluster, the objects within them are closer or more similar to one another than to objects in other clusters. This definition of closeness or similarity is given by a metric. Some examples of such a metric include Euclidean distance and Manhattan distance.

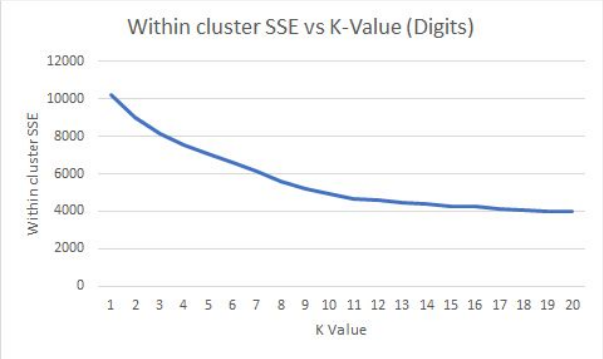
The similarity metric we have chosen to use for analysis of clustering algorithms is Euclidean distance, which uses a standard distance formula on attributes.

2.2 K-Means Clustering

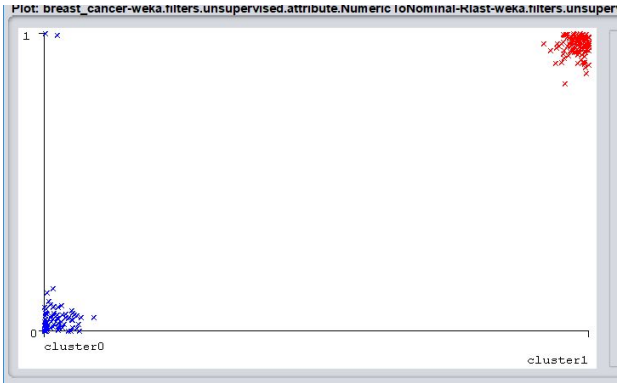
The first clustering algorithm we will look at is one of the simplest, yet still has more power than single linkage clustering. K-means clustering partitions the data into clusters in a more effective manner and produces more intuitive end results than single linkage clustering would be able to. K-means begins by picking k centers at random which do not necessarily have to be points of data (although it typically works better when they are). Each center then claims all the points of data that are closer to it than to other centers. For each of these centers, the center point is then recomputed based on the average location of all its cluster points. The algorithm then continues claiming points based on similarity or closeness based on these recomputed center points. This process continues until convergence, or when the center points no longer move.

With K-means clustering, the only hyperparameter that we can attempt to optimize on our dataset is the value of k, which corresponds to the target number of clusters. We then attempt to find this optimal number of clusters for our data:

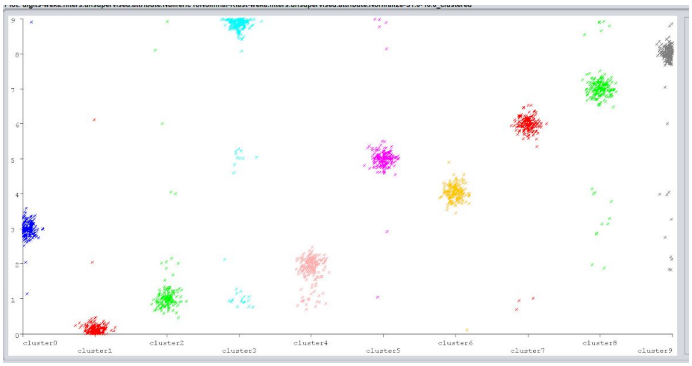




The within cluster SSE refers to the sum of squared errors within each cluster that the algorithm assigns (i.e. the actual Euclidean distance from each data point in a cluster to that cluster's centroid, averaged). To find the optimal number of clusters for our datasets, we used the elbow method, originally used by Robert L. Thorndike. Through the elbow method, we try to distinguish a clear point at which the error no longer decreases significantly (note: if we let k go on, it would decrease towards an error of 0 when the value of k would equal the number of data points). For the breast cancer dataset, the elbow point was determined to be k = 2, which makes sense given that there are two classes we are trying to classify: benign and malignant breast cancer cells. For the digits dataset, it was a bit more difficult to see the elbow point, but it seemed to be somewhere around k = 10. This also makes sense, as we also have 10 classes, one for each digit from 0 to 9. The within cluster SSE seems to be much higher than that of the breast cancer one due the complexity of the dataset. We have multiple handwritten digits from different people, so there will be a large variance between how people write their numbers.



This is a visualization of the breast cancer dataset with K-means clustering and a k value of 2. Its clusters are very good, as each data point in the cluster is very close. It also splits it into 2 classes resembling what the real life scenario is, with only a few errors for cluster 0.



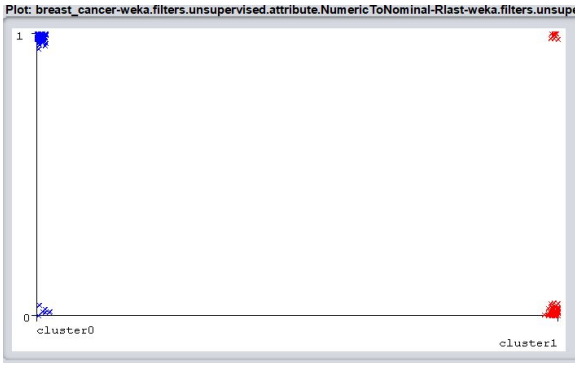
This is the visualization for the digits dataset. As shown, most of the clusters categorize the data very well. However, there is much more variance within the clusters, with the 9 digit having a lot of observations that are actually 1 digits and the 8 cluster having a lot of observations spread throughout multiple other digit designations. This is because of the similarity between some of the digits, which leads to some ambiguity for the algorithm and thus produces higher variances within clusters.

For both of these datasets, there are many other hyperparameters we could have tweaked in order to improve performance, including maxIterations and picking different starting centers. The centers are very crucial to finding the optimal centers because if we pick bad ones, there is a high chance of getting stuck at local optima.

2.3 Expectation Maximization (EM)

Unlike K-means, EM is a soft clustering algorithm. By using probability, it effectively allows a point to be in potentially as many clusters as possible (as opposed to a hard clustering, where a point is in a cluster or not). Thus, for expectation maximization, a point is not considered to completely be in one cluster, but for some k number of clusters, we assign to each point k probabilities, which signify the probability of that point being in a certain cluster. This in effect corresponds to a Gaussian distribution for each cluster, and each probability represents the chance that that particular distribution could've generated such a point. We then try to maximize the likelihood that the data could've come from these distributions.

To start off, we ran WEKA's expectation maximization clustering algorithm on both datasets, first allowing WEKA to first determine what the optimal number of clusters was before trying the optimal number that we found from our K-means algorithm. For our breast cancer dataset, WEKA selected k = 13, with a log likelihood of 34.8918. Using k = 2 from the K-means algorithm, we get that the log likelihood is 27.01516. We could select the value of k for EM by simply picking the value with the higher log likelihood, but given that the log likelihoods for both k values are so close, k = 2 may be the better choice. K = 13 is far more likely to have a higher log likelihood as a result of overfitting.

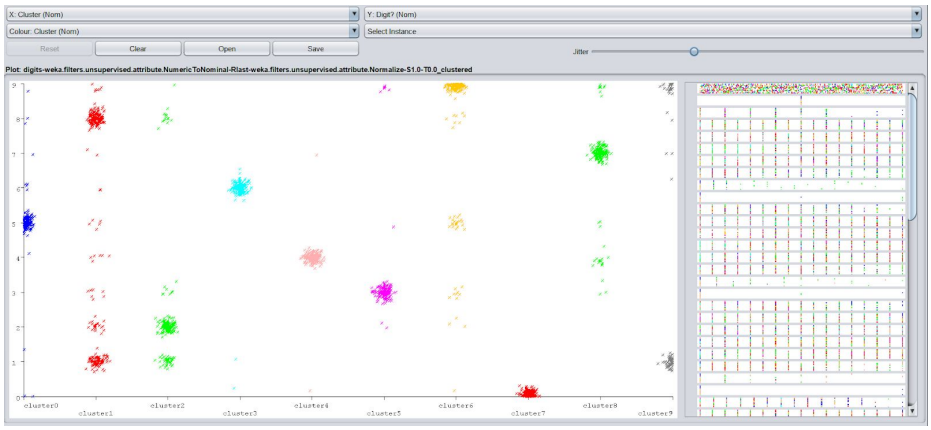


Breast Cancer Cluster Visualization (EM, k = 2)

Number of Clusters	Log Likelihood	Time to build model

2	27.01516	0.01 seconds
13	34.8918	16.12 seconds

These clusters are not as good as those that we created through K-means, as we see a lot more variance within each cluster. For the most part, in cluster 1, most of its values are gathered around 0, but there are a good amount that are actually categorized around 1. However, this visualization isn't able to show some of the "soft clustering", which is the strong point of Expectation Maximization.



Digits Cluster Visualization (EM, k = 10)

Number of Clusters	**Log Likelihood**	**Time to build model**
10	81.62573	0.85 seconds
13	85.06742	78.93 seconds

For the digits dataset, the selected k value was 13, and the log likelihood was 85.06742. However, with k = 10, the log likelihood was 81.62573, which is close enough such that we can use this as our optimal hyperparameter (since k = 13 is more likely to be overfitting). The clusters themselves are not that good given how spread out they are among the numerous digits, but again we do not have the soft clusterings that the program may be making. Thus, the algorithm will have digits that it's sure belong in one cluster, but others that it may be more unsure of as to whether it fits in. As this algorithm can get stuck, we can improve it by giving it more iterations and doing random restarts, just as we did with hill climbing. This also works with any distribution, so we could experiment with more than the Gaussian distribution as well.

2.4 Conclusion

Here is a summary of all the optimized hyperparameter values:

Dataset	**Clustering Algorithm**	**K**	**Misclassification Rate**
Breast Cancer	K-means	K = 2	7.21%
Breast Cancer	Expectation Maximization	K = 10	8.79%
Digits	K-means	K = 2	22.48%
Digits	Expectation Maximization	K = 10	24.99%

For both of these datasets, K-means performed better. One explanation for this could be that K-means was better at classifying due to the difference between hard clustering and soft clustering. K-means may be able to perform better when there are already disjoint classes within the data as the hard clusters are more likely to divide points of different classes, whereas for soft clustering, more points of different classes are more likely to end up in the same cluster. Overall, expectation maximization also takes much more time, possibly due to having to re-calculate each eigenvector and recompute the soft clusterings each iteration.

3 Dimensionality Reduction

3.1 Introduction to Dimensionality Reduction

While clustering revolves around finding some hidden structure of the data by grouping the data into partitions, dimensionality reduction attempts to reduce the amount of features or attributes required by a learning algorithm, in order to mitigate the curse of dimensionality. This improves the performance of the algorithm in both runtime and in accuracy. There are multiple ways of doing this, but they are normally categorized into two general ideas: feature selection and feature transformation.

Feature selection is the process of finding the best subset of actual features that still preserves (and possibly increases) the accuracy of a learning algorithm upon them. Some common methods include forward selection, whereby you iteratively add features until accuracy no longer improves, and backward selection, which involves taking away features that are not significant to the accuracy of the model.

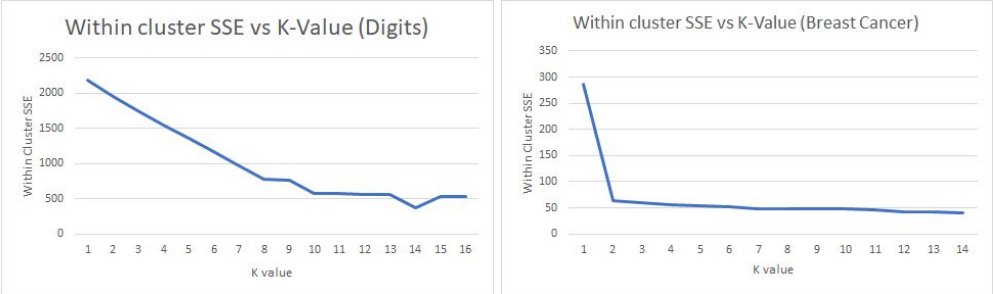
On the other hand, feature transformation involves constructing transformed versions of existing features to preserve and potentially increase accuracy. This can be done by determining a transformation of the features into some smaller feature space, projections onto which will allow us to achieve similar if not better accuracy than on the original dataset. This transformation typically reduces the number of features, thereby reducing the curse of dimensionality. This may make the data linearly separable, similar to the way the kernel trick works with support vector machines (SVMs). This gives feature transformation more power than feature selection. Thus, we will be analyzing its effectiveness with four algorithms.

3.2 Principal Component Analysis (PCA)

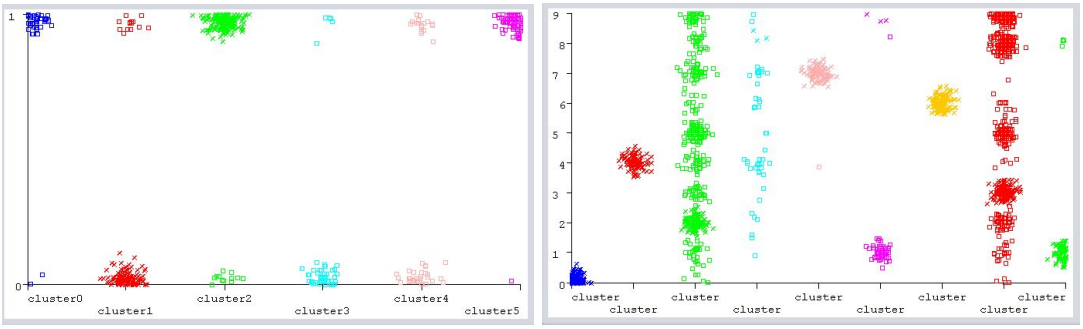
Principal component analysis (PCA) is a feature transformation algorithm that interprets the problem as an eigenproblem and attempts to find a linear transformation of the features such that the projections of the data onto the transformed axes have maximum variance. These transformed axes are known as the principal component vectors. After finding a first principal component vector, we then look for orthogonal principal component vectors to add more dimensions with which the data is modeled. PCA performs well if it can reduce the number of features down considerably; this would be considered good performance for the algorithm. Here, we are attempting to minimize reconstruction error, which is the sum of distances from the axis of the principal component on which the points are projected onto.

Using WEKA's PCA algorithm, 10 principal components passed the criterion for the breast cancer data and 40 principal component vectors for the digits data. Compared to the original 30 and 64 attributes respectively, PCA helped significantly reduce the number of features and thus the curse of dimensionality. This especially makes sense for the breast cancer data, as a lot of the data was composed by some linear combination of 10 features of the cancer cells. The digits dataset was a little more difficult to interpret as each input is a cell of an 8x8 input matrix. Four principal components were necessary to explain 80% of the variation for the breast cancer set (the first principal

component explained 44%), while for the digits dataset, the first component only explains 12% and 20 components are necessary to explain 80% of the variation. It seems that the reduction in dimensionality needs to be balanced, so I ran the clustering algorithms once more to see how the results might change.



For K-means, the optimal k value found for the breast cancer dataset was 2, and the k value for digits was 10. This K-means cluster for the breast cancer data had a misclassification rate of 8.26%, which is worse than before PCA. The K-means cluster for the digits dataset had a misclassification rate of 40.12%, which is also significantly worse than before. Thus, we most likely got rid of important information through dimensionality reduction. Next, we tried expectation maximization:



Left: EM with k = 6 (Breast Cancer); Right: EM with k = 9

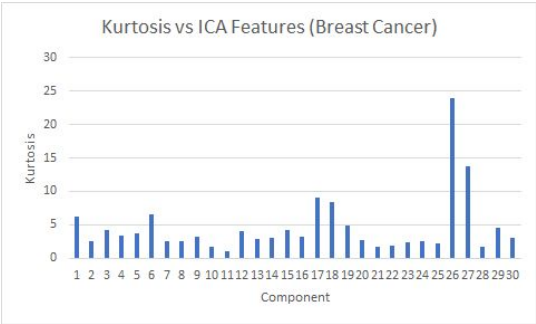
These K values were chosen by default by WEKA. From looking at these clusters, it appears that EM also performed worse than before, especially for the digits dataset where some of the clusters aren't even separated well at all. However, this makes sense, as the original dataset was already a reduction of 32x32 bitmaps to 8x8. This means that we had too much dimensionality reduction, which led to a loss of information.

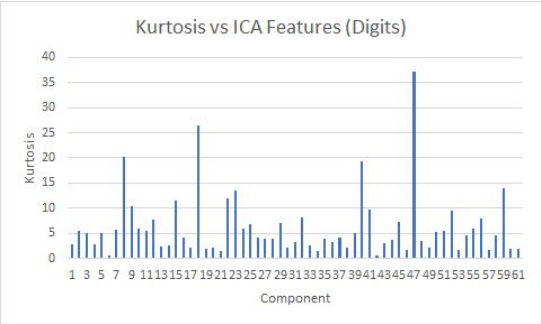
3.3 Independent Component Analysis (ICA)

Independent component analysis is another dimensionality reduction algorithm that has similar goals to principal component analysis but accomplishes those by turning the original feature space into one such that each feature is independent. It works by assuming there are hidden independent features underlying a dataset's features and tries to find these independent components. These components shouldn't provide any mutual information to each other, but when considered as a whole, they maximize mutual information and allow us to potentially reconstruct our original dataset.

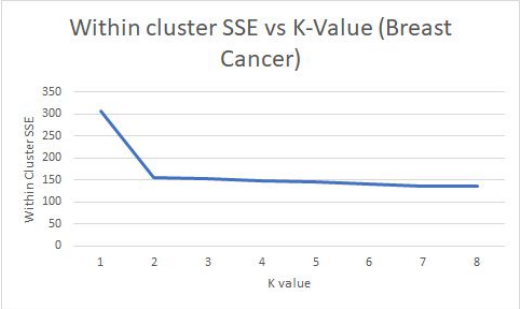
We then ran the ICA algorithm on each dataset, computing the independent components for each dataset. Since the ICA assumes that each independent component is highly non-normal, we analyzed each ICA components' kurtosis

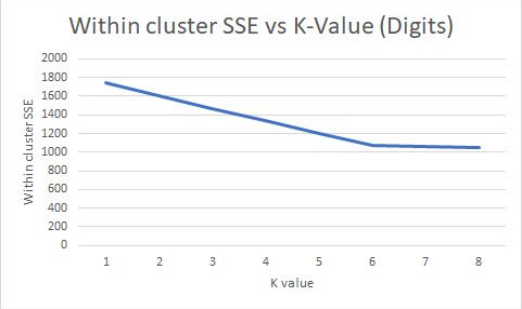
value to ensure that they were non-normal. A kurtosis of 3 would indicate an ideal normal distribution, so if the kurtoses of these ICA components are far from that, then we can determine how successful ICA was.



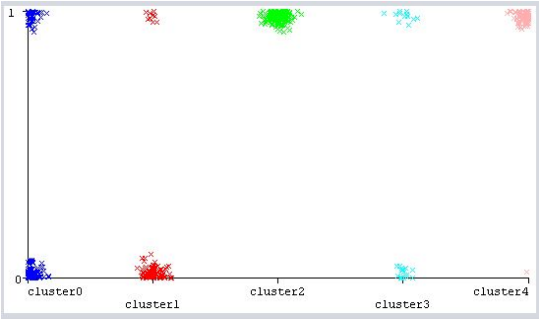


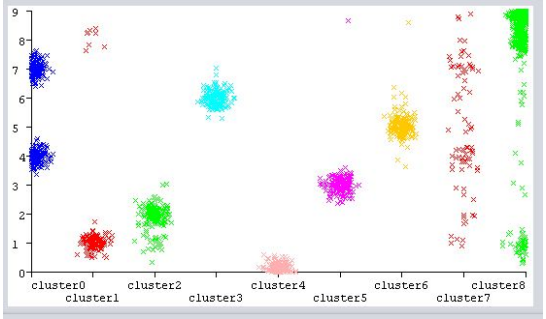
There are a few features in each dataset that kurtoses hinting at a normal distribution, but for the most part there is a high degree of mutual independence among the generated features. Because of this, I continued on to run the clustering algorithms on the ICA generated data using the same k values from PCA.





Interestingly, the optimal k-value from the elbow method for the breast cancer stayed the same at k = 2, while the optimal k-value for the digits dataset changed to be k = 6. This could be due to the breast cancer dataset having more disjoint classes.





Left: EM with k = 6 (Breast Cancer); Right: EM with k = 9

The clusters were worse overall compared to PCA, which was already worse than the original dataset. The breast cancer dataset had too much separation across 1s and 0s (although it had one good cluster), and the digits dataset had a lot of observations that were more spread out due to having more classes. This may be due to the ICA components

not actually being non-Gaussian and that the features are not mutually independent. A lot of the features in the breast cancer dataset are also not independent (as they are derived from each other), so that might have affected the the success of ICA.

3.4 Randomized Projections (RP)

While PCA and ICA are quite powerful algorithms, RP runs significantly faster and works well enough in practice. This algorithm takes a specified target number of dimensions and uses that information to generate a randomized matrix that is then used to transform and project the features into another feature vector in a smaller feature space of the specified number of dimensions. Since randomized projection starts off by generating the random matrix, the way this matrix is created plays an important role in the success of RP. I also tinkered with the number of attributes (target dimensions) to see how this would affect the algorithm.

Breast Cancer dataset:

Clustering Algorithm	**Number of Attributes**	**Number of clusters**	**Log likelihood**
K-means	10	2	169.246765655109
K-means	15	2	92.2785109511794
K-means	20	2	120.6
EM	10	8	-7.53
EM	15	7	-10.83
EM	20	12	-12.64

Digits dataset:

Clustering Algorithm	**Number of Attributes**	**Number of clusters**	**Log likelihood**
K-means	10	9	192.41
K-means	15	9	303.929
K-means	20	9	426.628
EM	10	9	-19.567
EM	15	9	-30.16
EM	20	9	-38.21

There were very few patterns in the data obtained. Although log likelihood seemed to increase with number of attributes for the digits dataset, this did not occur for the breast cancer dataset (although this may be due to the number of features in the digits dataset). These erratic results may be explained by the nature of the algorithm, as it does rely on probability. The choice of the seed can place the algorithm in a place such that it will get stuck at a local optimum or end quickly. This goes to show that randomized projections is not a very reliable algorithm

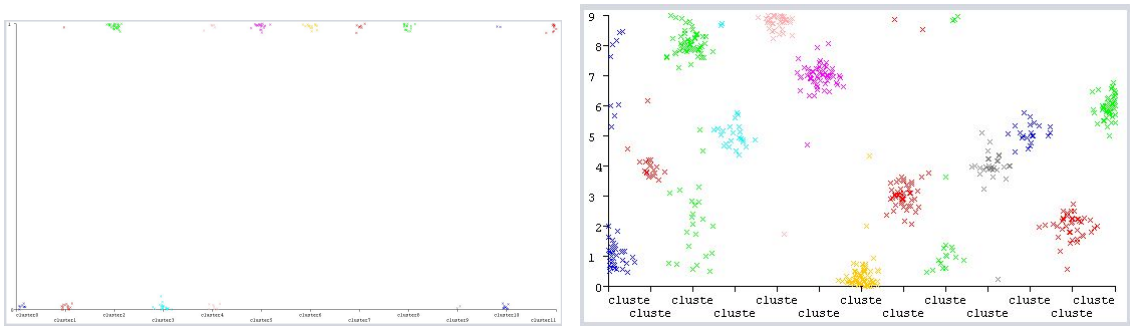
(variance in reconstruction error) unless we have more iterations to ensure that it has enough tries to find the global optimum.

3.5 Information Gain (IG)

The next algorithm that we will be analyzing is information gain, which is different from the rest of the three feature transformation algorithms that we've looked at in that it is a feature selection algorithm. This concept should be very familiar, as we used this with decision trees in supervised learning. ID3, the decision tree algorithm we used, selects attributes by maximizing information gain; information gain feature selection works exactly the same way.

Dataset	**Clustering Algorithm**	**Dropped Attributes**	**Number of Attributes**	**Log likelihood**	**Number of clusters**
Breast Cancer	K-means	4	26	143.87	2
Breast Cancer	EM	4	26	31.31724	12
Digits	K-means	9	55	3925.15	9
Digits	EM	9	55	29.17439	13

For these attributes, we dropped features by evaluating their average merit. If their confidence interval contained 0, the attribute was dropped. This led to 4 features being dropped for breast cancer and 9 for the digits dataset. Below are the EM visualizations for k = 12 (breast cancer) and k = 9 (breast cancer). Misclassification rates on k-means was 7.03% for breast cancer and 24.76% for digits, which are both lower than before removing the attributes.



Left: EM with k = 12 (Breast Cancer); Right: EM with k = 13

These clusters appear to be better than the original clusters in terms of actually dividing the data by the class labels. Overall, this algorithm showed similar performance to the other dimensionality reduction algorithms. The breast cancer dataset seems to be a bit better, and the digits dataset seems to still have some variance among its clusters.

3.5 Conclusion

ICA and IG seemed to be the most successful and had results that made the most sense.

4 Neural Network Learner Results

Next, we will evaluate the efficacy of dimensionality reduction and clustering on a neural network learner, with respect to the breast cancer dataset.

4.1 Dimensionality Reduction

Below is a table of our neural network results, using all our tuned hyperparameters and findings from above:

DR Algorithm	**Time to build**	**Misclassification rate**	**F1 score**
None	8.45	4.0422%	0.96
PCA	3.83	2.6362%	0.974
ICA	9.78	4.9209%	0.951
Randomized Projection	3.9	5.7996%	0.942
Information Gain	7.36	3.5149%	0.965

Interestingly, PCA and ICA performed the best (as opposed to ICA and IG before), as they were able to find attributes that impacted the learning the most and singled out attributes that were not as significant (e.g. smoothness error, texture error, and symmetry error). For algorithms such as IG, these features were dropped or just were not included in the components. Overall, these clusters seemed to be moderately better after applying the DR methods.

4.2 Clustering

In order to see if using clusters as features can help learn a neural network, WEKA has an AddCluster filter that finds clusters and assigns the cluster value as an attribute to every point.

Clustering Algorithm	**Time to build**	**Misclassification rate**	**F1 score**
None	8.45 seconds	4.0422%	0.96
K-means (k = 2)	8.3 seconds	4.2179%	0.958
EM (k = 14)	11.36 seconds	2.6362%	0.974

EM was took the most time to build but ended up being the most accurate. K-means was not as successful as the original because the clusters formed weren't the best at separating the two classes, whereas EM found 14 clusters that ended up having unique class types in each cluster. Overall, it seems that modifying the original dataset through combinations of feature transformation clustering can decrease the build time and increase the overall accuracy.