Clustering and Feature Reduction

OMSCS 7641 Machine Learning

Jeff McGehee

Introduction and Data Descriptions

In this paper, I will document my findings regarding clustering and feature reduction on the "Internet Advertisements" and "Heart Arrhythmia" datasets. Both of these datasets were outlined in assignment one, but in order to refresh the memory of the reader, I will give a short summary of the data as it relates to clustering and feature reduction. Both of the datasets have a very high dimensionality, which will make this exploration very meaningful in terms of supervised learner performance. "Internet Advertisements" is designed to classify html objects as "ad" or "non-ad", on the other hand, "Heart Arrhythmia" is designed to classify 13 different types of arrhythmia from ECG data.

"Heart Arrhythmia" consists of 279 input features and only 452 instances being spread over 13 classes. It can easily be seen that there are less than 2x the number of learning samples as there are input features, and this is is magnified even more when the set is split into training/testing sets. Similarly, but on a larger scale, "Internet Advertisements" consists of 1558 input features and 2359 instances being spread over just two classes. This means that in the current state for both datasets, Haussler's Theorem (Eq. 1) is unlikely to be satisfied with any desirable error, ε.

$$m \geq \frac{1}{\varepsilon}(ln|H| + ln\frac{1}{\delta}) \qquad \text{Eq.1}$$

While each dataset does have a large number of input features, I will note that this is largely due to the presence of many binary features. For "Heart Arrhythmia" these represent the presence of many artifacts in an ECG test. For "Internet Advertisement" set, this is due to one-hot encoding of text features for urls, etc.

Clustering Analysis

My datasets were chosen before I watched the lecture regarding the curse of dimensionality. Both of them contain a very large number of features, for this reason, it was somewhat challenging to determine ways to visualize the "correctness" of the clustering algorithms. In fact, for both clustering algorithms, it seemed necessary to use the knowledge of the dataset classes as a measure of clustering performance.

K Means Clustering

To choose K, it seems natural (since I am interested in improving my supervised learning performance) to choose K equal to the number of output classes for the given dataset. However, since I am still a *student* in machine learning, I decided I should try other values of K as well to determine if they may be more valuable. Because of the extreme dimensionality of my datasets, I looked for the average distance from each point to its respective cluster center (called inertia in the sci-kit learn implementation) in order to have some understanding of how the algorithm was performing. Because K means does not necessarily converge to the same solution each time, this was done 50 times for each K value. The resulting averages and standard deviations are shown in the bar charts below.

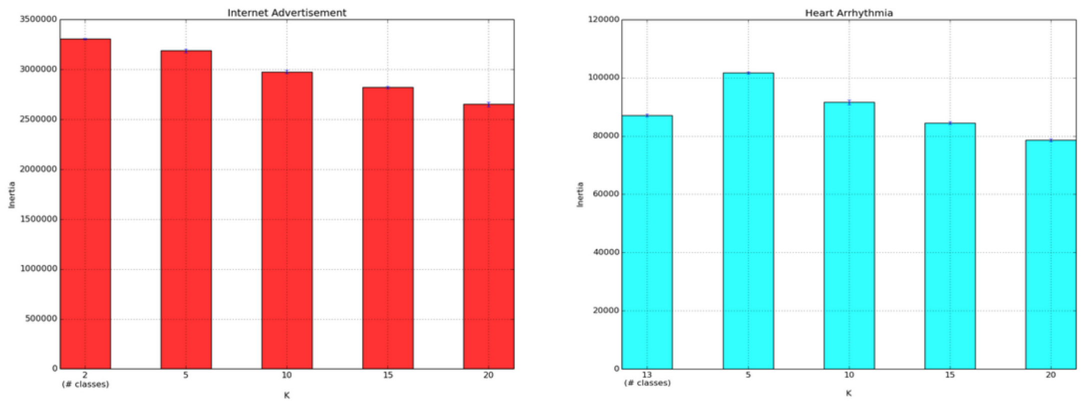
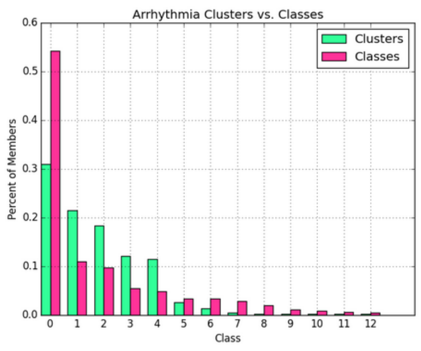
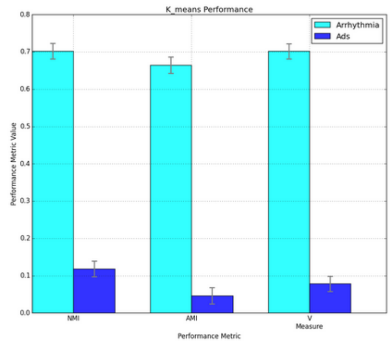


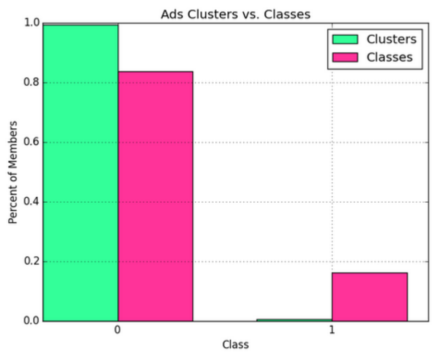
Figure 1: ***Trying to Select K***

What I gathered from this analysis is that the inertia seems to simply be decreasing linearly in K, meaning that for the values tested, none seem to fit the data extremely better than the others. Also, note that the standard deviation for each value is extremely small, meaning that K-means generally converges to similar solutions each time for every value of K (for these datasets). Based on this, it seems ideal to pick K equal to each dataset's number of output classes, since no other K seems to provide better results, and more meaningful info can be gathered if K matches the output classes.

With K equal to the number of output classes, we can look at how the clusters compare to the output classes. The results of this can be seen in the figure below.



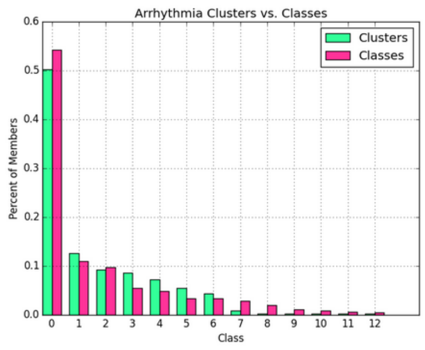


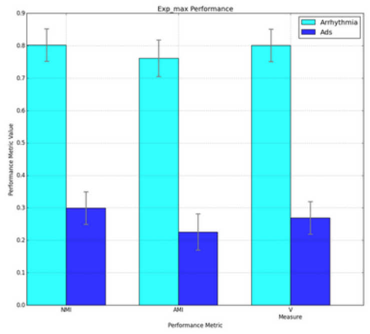


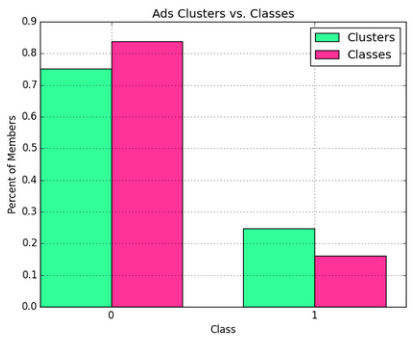
Because the clustering algorithm is unsupervised, and my datasets have extremely high dimensionality, it is difficult to directly relate the cluster numbers produced by the algorithm to the proper output class. To get the best idea about this that I could, I compared the sorted (by number of members) clusters vs. the sorted class sizes. The two outside figures show the actual cluster/class relationships for a single typical run of K-means. The middle figure shows performance metrics (AMI, NMI, and V-measure) for both datasets over 50 runs of K-means.

Expectation Maximization

To determine K, I decided to look at the highest clustering probability for each point and select a K value that has the least amount of "in-between" points. I chose this method because this paper is focused on clustering as it relates to classification, so I want definitive clusters that can be used to aid in classification. **It was found for both datasets, for all tested instances of K (same as above), every point had a probability of belonging to its cluster of greater than .99. Therefore, as with K-means, I chose a K value equal to the number of output classes for each dataset.**







Looking at the figures above, it can be seen that the performance is better than K-means for both datasets. These figures were generated in the same manner as the figures in the K-means section.

Clustering Summary

For both of my datasets, Expectation Maximization seemed to perform best, and out of my two datasets, the "Arrhythmia" was able to be clustered in a fashion that most matched its output classes. I suspect the poorer performance of the "Advertisement" dataset is due to the fact that there may be many more cluster-able groups in the highly dimensional space than there are output classes. Since I am asking the algorithms to only find two clusters, it is almost certain that the "ideal" two clusters that represent the data classes are swallowed by larger clusters.

Though EM was the best performer on these datasets, it does have its weaknesses. EM relies on the selection of a good starting point and can "get stuck". This can be solved by random restart (which I implemented for my simulations), but it would be interesting to build some random optimization into the algorithm by combining it with SA for example to randomly explore different cluster centers.

K means also suffers from the same "getting stuck" problem of EM, so it would be neat to include some randomized optimization features to avoid this. Also, K means, because it is computing averages, can be heavily influenced by outliers. It would be good to avoid this by computing weighted means.

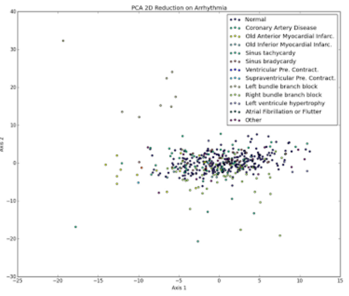
It would also be interesting to explore making changes to both algorithms that would avoid an apriori selection of K. Again, this could be done with some sort of randomized optimization algorithm where the fitness is some domain knowledge known about the dataset. The algorithm could then choose an optimal K based on this fitness. I'm not sure if it necessarily qualifies to a "change" to the algorithms, but it would also be advantageous to use the most relevant distance computation for the given data domain.

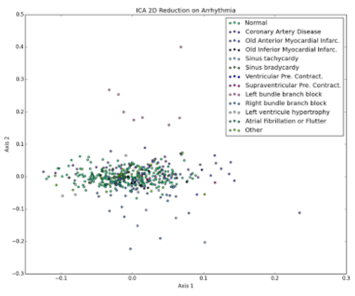
Dimensionality Reduction

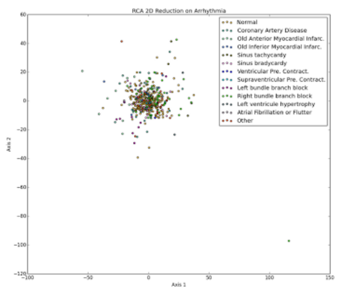
The high dimensionality of my datasets makes them ideal candidates for dimensionality reduction. To get an understanding of how the dimensionality reduction algorithms (PCA, ICA, RCA, LDA) perform on the data, I first looked at how they performed when reducing the data down to 2 and 3 dimensions. For both datasets, the behavior in 3 dimensions was very similar to the 2D reduction, so for ease of presentation, all data shown is 2D. These results for the “Heart Arrhythmia” data set can be seen in the figures below. **It should be noted that for all scatter plots show, multiple runs were performed for all of the reduction algorithms and the plots shown are representative of the overall performance seen over the multiple trials.**

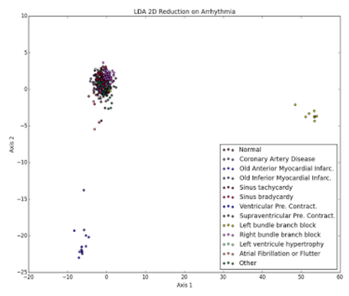
The worst performer here is RCA. Upon inspection of the data, there is no class of points that are standing out from the rest. This is probably because it is randomly choosing only two components out of over 250 dimensions. However, one thing about RCA and random selection in general is that as we allow it to pick more components, the likelihood of it picking the best ones will increase.

On the opposite end of the spectrum is LDA. This is not a surprise because LDA is somewhat of a “supervised” learner, which has already been given correct class labels. However, in assignment 1, all of my supervised learners performed poorly on this dataset, so there must be more going on here. We see that there are two classes very clearly separated from the group. LDA chooses (in this case 2) Linear Discriminants that maximize class separations, and the fact that this is done very well here, suggests that there is little or no correlation between features. This lack of correlation between features most likely confused other algorithms, but it works in favor of LDA.

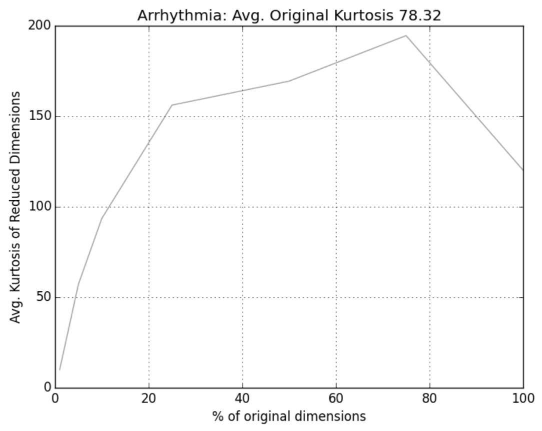


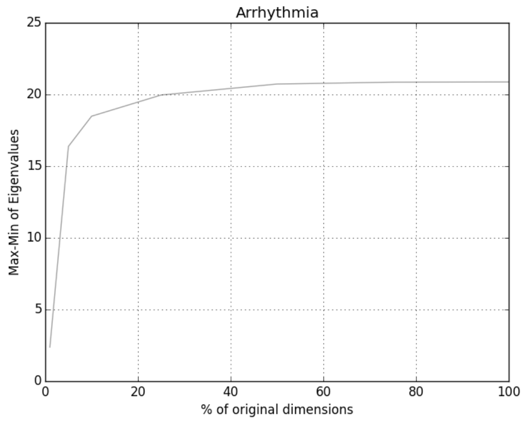




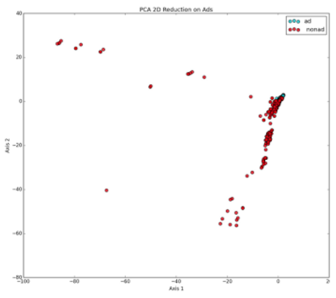


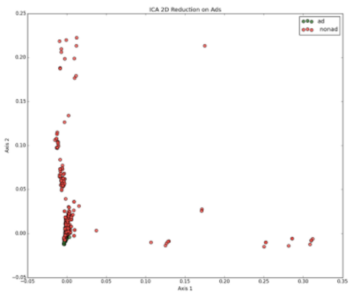
Somewhere in the middle are PCA and ICA. PCA and ICA perform similarly, mostly showing the data as what appears on first glance to be an elongated “blob”. However, if you look more closely, the points scattered away on the outside of these blobs all belong to two (for a 2D reduction) classes. This gives me an indication that to some extent, PCA and ICA are working at increasing the variance in the data. Below we can see the eigenvalues and kurtosis information for PCA and ICA, respectively. For both metrics, I chose to plot the data with respect to the percentage of the the original dimensionality of the data, so the number of features increase from left to right. On the left is the average kurtosis of the features for ICA(the average kurtosis of the original dataset can be seen in the figure title). On the right is the absolute value of the difference in the eigenvalues. Here, a high value means the first principal component has much more variance than the remainder of the principal components.

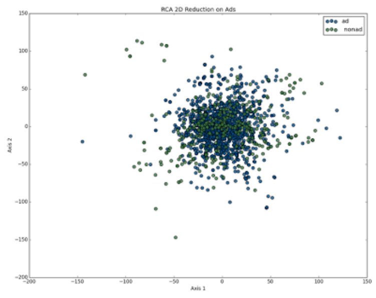


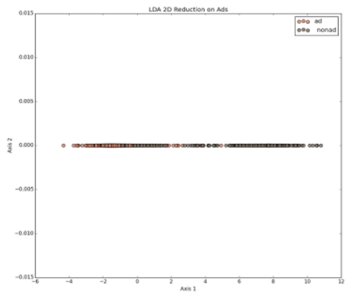


The information for the "Advertisement" dataset can be seen in the next set of figures. Again, RCA appears to have trouble choosing two random components that are meaningful for this very high dimensionality dataset. LDA was unable to produce a 2D reduction because it is only able create at most one less feature than the number of classes, which for the "Advertisement" dataset is 2. However, it does appear to do a fairly good job at separating the two classes.

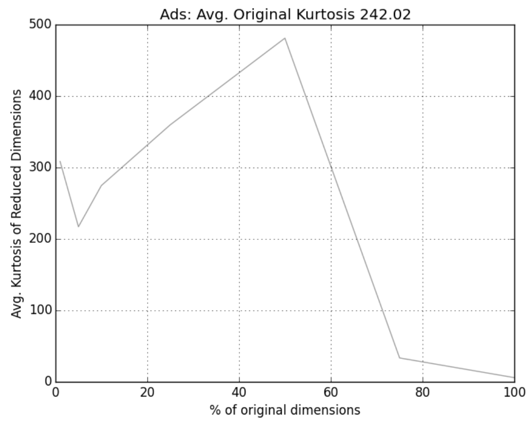


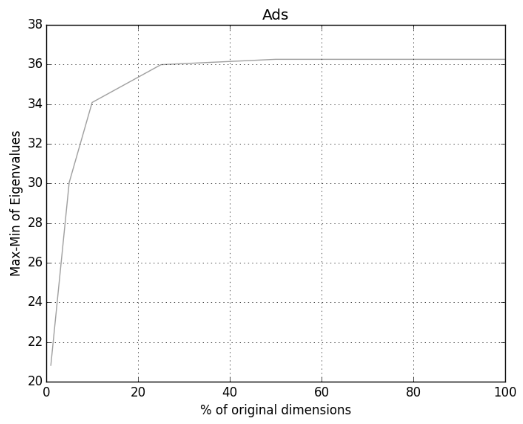






PCA and ICA both find similar components to add variance to the data, as can be seen by their similar graphs. For both algorithms, the "ad" class remains clustered together near the "point" of the scatter plot, so possibly as we increase the the number of principle/independent components, the remainder of the "non-ad" points would be spread away from the "ad" points. I also noticed that both algorithms bring out groupings of the data that are unrelated to the class of the data. This suggests that there are some "irrelevant" features in the datasets. The eigenvalue and kurtosis information can be seen below. The plots contain the same information as above, but for the "Advertisement" dataset.

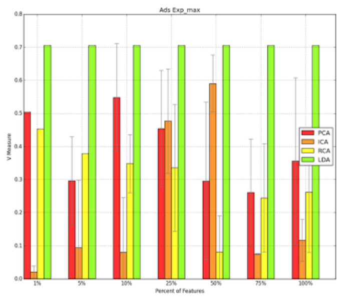


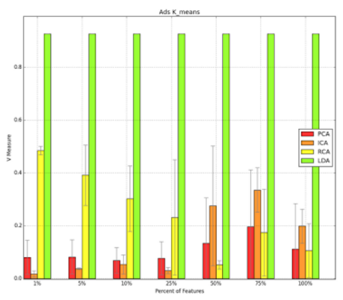


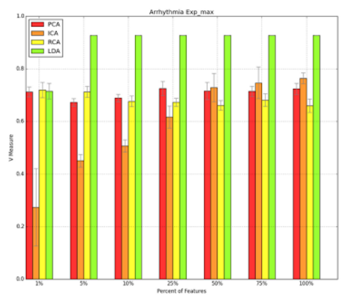
The eigenvalue plots for “Advertisement” and “Arrhythmia” indicate that there are less principal components in the “Advertisement” because the difference between the eigenvectors increases more rapidly and converges at a higher value than for “Arrhythmia”.

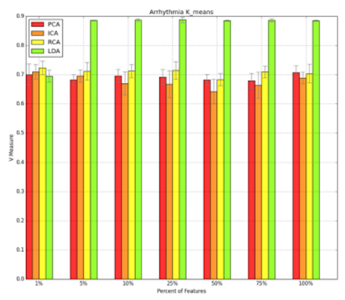
Combining Clustering and Dimensionality Reduction

Since it has been understood how the various dimensionality reduction algorithms perform on the datasets, it is intuitive to see how they influence the clustering performance on the data. In order to do this, I built a simulation to calculate the V-measure of the output of both clustering algorithms after they have been given data from each reduction algorithm. This was was then done for various “levels” of reduction. The x axis in the figures below represents the number of features in terms of the percent of the original dataset, and the y axis is the V-measure.









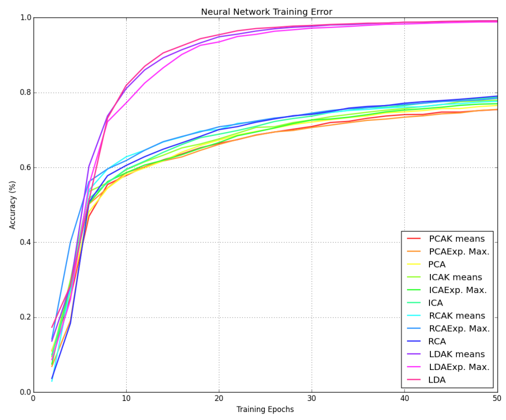
As can be seen by looking at the V-measure for the cluster outputs versus the class data, for PCA, ICA, and RCA, the clusters are very similar for the “Arrhythmia” set, but radically different on “Advertisement”. LDA, however, is able to produce clusterings that are much more similar to the data consistently for both datasets. This is because LDA is given class labels and attempts to maximize the separation between datapoints in a way that matches their class labels. This obviously boosts its performance on a test where I am comparing clusters with output labels. I also found it interesting that though RCA is “random” its variance in performance is quite similar to the other algorithms.

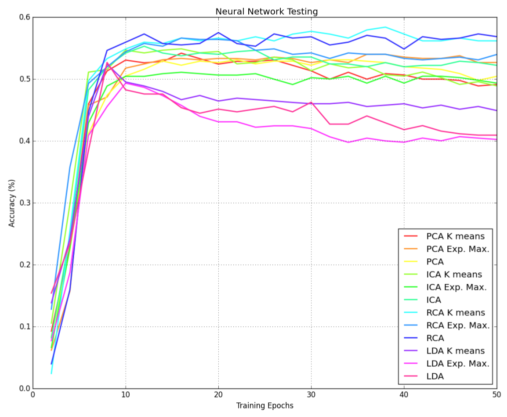
The results of this test are similar to the original clustering experiments, in terms of V-measure of the “Arrhythmia” data versus that of the “Advertisement” data as well as K-means versus EM. However, there is a positive shift for all. With the “Arrhythmia” dataset, this is consistent, but for “Advertisements” there is a wild variation in performance. This is likely due to the previously mentioned problem of trying to produce only two clusters with the “Advertisement” set.

Applications in Supervised Learning

Now that I’ve found how the various feature reduction algorithms behave on each dataset, it makes sense to apply the feature reduction algorithms to a supervised learning problem. The “heart arrythmia” dataset has proven to be the most difficult to learn in all assignments so far, so I chose it as the candidate to learn using feature reduction. I suspected that since the feature reduction algorithms appeared to do well at “accurately” reproducing the data, feature reductions would solve the problems that supervised learners have had with this dataset. However, I learned that this was not the case because of a few main factors. I will show why and how through simulations performed with a neural network learner.

To see how all of the feature reduction techniques affect the neural network learner, I ran a simulation to investigate the performance of each combination of reducer/clusterer (including just a reducer on its own), on a 90% - 10% training/test split of the “heart arrhythmia” data. The results can be seen below:

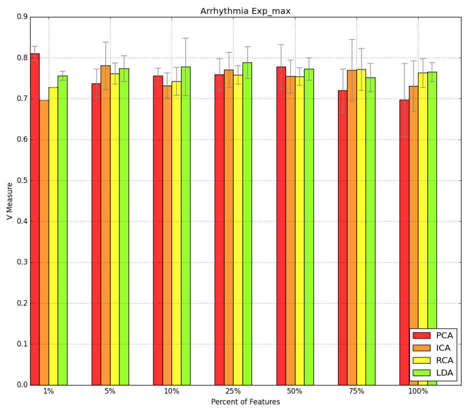


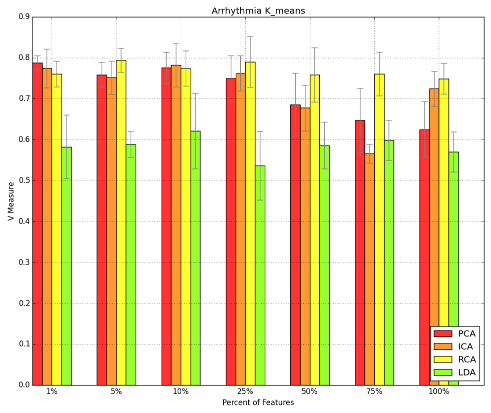


First, I noticed that the training error patterns follow the same pattern as the performance of the reduction algorithms on the "arrhythmia" dataset. LDA is the best performer, followed by ICA, while the others all perform similarly. Second, it seems that the presence of a "cluster" feature added onto the reduced dataset did not make a big difference in the training or testing error. Comparing to the performance seen in assignment 1, the reduction and clustering algorithms do help with training error (in the case of LDA and ICA), and there was a significant boost in speed due to the reduced size of the dataset (and hence less weights to optimize each iteration). However, there is not much of a change in testing error from assignment 1, and interestingly, the best training performer (LDA) has the worst testing error. Like in assignment one, there is not much evidence of overfitting.

The lack of overfitting implies that still, there are not enough training samples for the learner to learn the large number of classes, and the dataset has high variance. Looking at the clustering figures, and recalling from my analysis in assignment 1, over 50% of the "arrhythmia" dataset is one class, and the remaining samples are divided (non-uniformly) among the other 12 classes. This makes it very probable that a training set could be missing examples of multiple classes entirely. This cannot be avoided with feature reduction, although as I am writing this, I realize that it may be possible to get better testing performance if the testing and training sets were manually selected to include all classes. I am frustrated that I am just now coming to this conclusion and do not have time to run a test.

To investigate why LDA performs so well on training, but has no improvement in testing, I reproduced the experiment performed in "Combining Clustering and Dimensionality Reduction", except I used a training set of my data to "fit" the reduction algorithms, then used these models to transform a corresponding test set.





It can be seen that though LDA outperformed (in terms of making clusters match classes) the other algorithms when it uses all the data, it's performance is similar to the others (or slightly worse) when it is asked to reduce data not used to build the model. This is because, unlike the other reduction algorithms, LDA relies on the class labels to build its transformation model. Since the model was built only on the training data, which as mentioned above may not include some classes, LDA's performance is affected. Because PCA, ICA, and RCA do not rely on class labels, they seem to perform similar to before.

In conclusion, it seems that dimensionality reduction was not able to improve the testing classification performance of the ANN on the "Arrhythmia" dataset. The seemingly excellent performance of LDA was also proven to fail when asked to handle a part of the "Arrhythmia" dataset it had not seen before. Though I was hopeful that the high dimensionality of "Arrhythmia" was to blame for its difficulty so far, it seems that without a very calculated train/test split as mentioned previously, the "Arrhythmia" dataset simply does not contain enough examples of each class to be reliably predicted by a supervised classifier.