Assignment 3 - Unsupervised Learning

Jon Worley

March 20, 2015

Introduction

This project in CS 7641 is aimed at exploring clustering and feature transformation algorithms, and introducing the students to unsupervised learning techniques. These techniques are useful when your dataset does not have a "Class" column at the end of it. The goal of unsupervised learning is to figure out how much you can know about the data given to you. It turns out, that there are a lot of ways that you can modify data, and turn it into useful information that could easily be used for a person to make decisions, but it also can be used to feed a ML program so that it can also "learn" to make decisions

Dataset Discussion

For this project, I attempted to use three datasets to explore clustering and feature transformation techniques. I used the two datasets that I had been using for previous projects, the Breast-Cancer and Glass datasets, but I also introduced the Iris dataset mostly so that I could really start at the basics of analysis and work up from there

Iris

In previous projects, I had resisted using the Iris dataset because I thought it would be too simple for analysis. In previous projects, though, I was finding a lot of difficulty in using more complicated datasets right away. I decided to take a step back, use the simpler dataset at the beginning so that I could familiarize myself with the necessary code for the project, and then use the more complicated datasets for the project.

What I found was that the Iris dataset is much more interesting than I originally thought. It is simple, but it is simple in a way that allows the user to get a much better understanding of how all the algorithms work. It is a very approachable dataset it is a very graph-able dataset, and it is a very malleable dataset. My other two datasets are interesting because of their unique properties, but it is much easier to "black-box" the more complicated datasets, and not get a deep understanding of what is going on. The Iris dataset lays everything out there for you to gain a real appreciation of the algorithm.

Breast Cancer

The breast cancer dataset is a very interesting dataset. Not only has it been very hard for algorithms to perform well on, it is also almost completely categorical. Trying to use this dataset with algorithms that are rooted in linear algebra concepts presents an interesting problem. Since readily available R algorithms don't have the ability to handle even factored categorical datasets, I found myself turning the breast cancer dataset from a 10-20 feature categorical dataset into a 50-100 feature binary dataset. This presents an interesting problem for not only distance calculations but also feature transformations

Glass

The glass dataset is simply interesting because it contains 7 possible classifications rather than the standard 2.

Methods

K

For this project, all of my datasets had known classifications that I had to remove to turn them into unsupervised sets. Since I already knew the number of classifications, I simply used that number for K. I did experiment with choosing other K values, but those were fairly arbitrary choices and resulted in expected poor performance. There are algorithms that I found that will search for appropriate K values, but I didn't include them into this analysis for sake of brevity.

Distance and Similarity

For the Iris and Glass datasets, Euclidean distance is a sufficient metric of similarity since the features all exist on in a continuous space.

Finding a metric of similarity for the categorical breast cancer dataset was a little more difficult. Most algorithms weren't able to accept a categorical dataset, so I converted the data into binary bit strings with many more features. I had hoped that using a hamming distance metric would help categorize the data better, but I ended up only using algorithms that could accept the categorical string data like k-mode and expectation maximization.

Clusters

For the datasets that exist in a continuous space, the clusters are exactly what you would expect them to be. in and two dimensions, they are clouds of data points that are close to each other. The number of "clouds" is exactly equal to the K value that is set at the beginning of analysis. These clouds are very similar to KNN types of analysis (especially when using Euclidean distance).

For datasets that don't exist in continuous space, the clusters are a little more visually ambiguous. This is simply because there aren't as many geometric shapes that you can draw around the clusters to represent the cluster shape. These clusters are better represented by analyzing their results and accuracy rather than the

Feature Transformation and Cluster Spaces

Feature transformations vary widely in being able to visual the transformation that has taken place and also understanding why the data and the resulting cluster has the new shape. The new cluster shapes happen because the data has actually been modified by some real amount.

The easiest transformation to understand is filtering technique. That technique simply decides to not use a feature that a given algorithm has determined is unimportant. The dimension reduction occurs by removing dimensions.

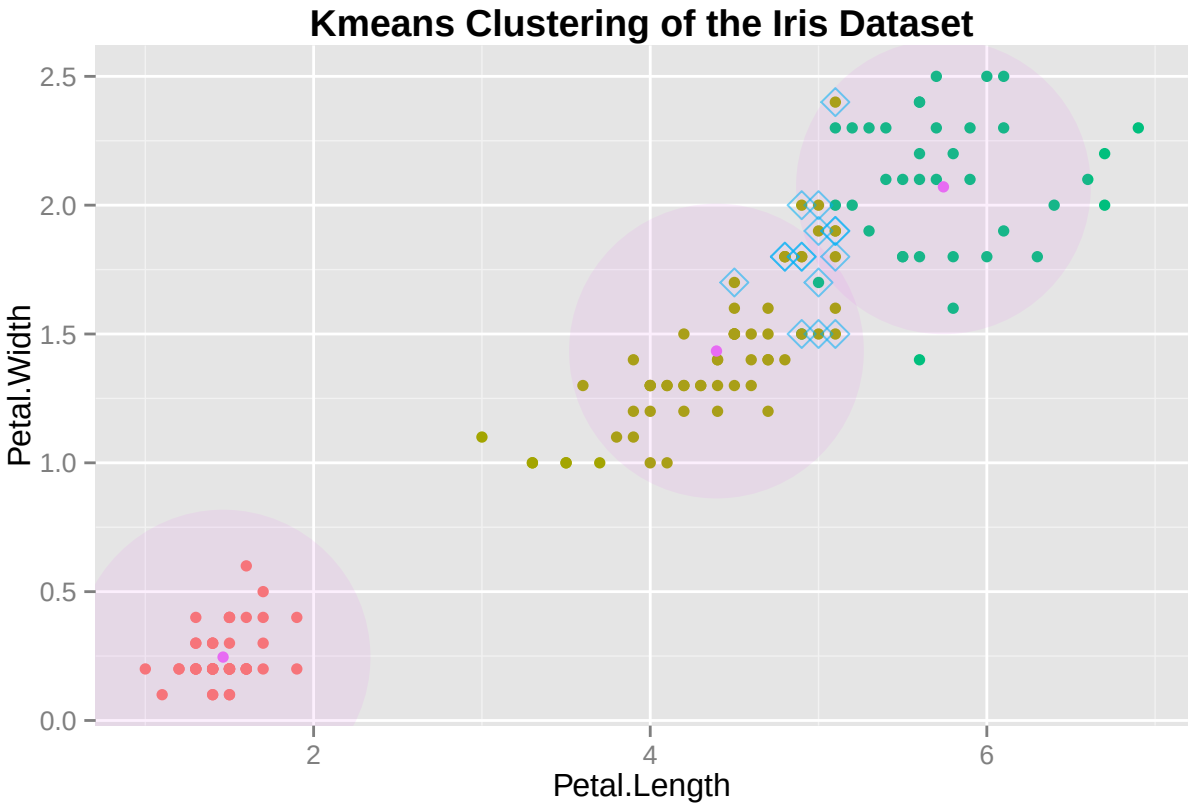
The algorithms that are a little more complicated than filtering is PCA and RCA these algorithms can be viewed as a data rotation around a new axis. PCA and RCA have different ways of determining how the data is rotated, but they essentially project the data into the same space. Both algorithms are projecting the data into a new space by using a compression algorithm to turn multiple feature dimensions into a reduced dimensional space. The benefit of PCA is that you don't actually lose any data when you do the compression due to the requirement that PCA maintains orthagonality with the new dimensions. RCA has the potential to compress the dataset and lose a high amount of information depending on how large your original n features are, and how small the new m features are. You will lose more information and the difference between m and n increases.

PCA has a lot of theoretical advantages, but it doesn't always allow for easier analysis because of this compression, though. I view the algorithm as squashing high dimensional data into a low space. This is great for computation complexity, but may have the effect of reducing distances of two data points that should be classified differently. If the distances are reduced too much, you may set a limit on how well you can classify data. Depending on the data, though, the dimension compression can create features that are more important that any of the previous features and perform both accurately and quickly

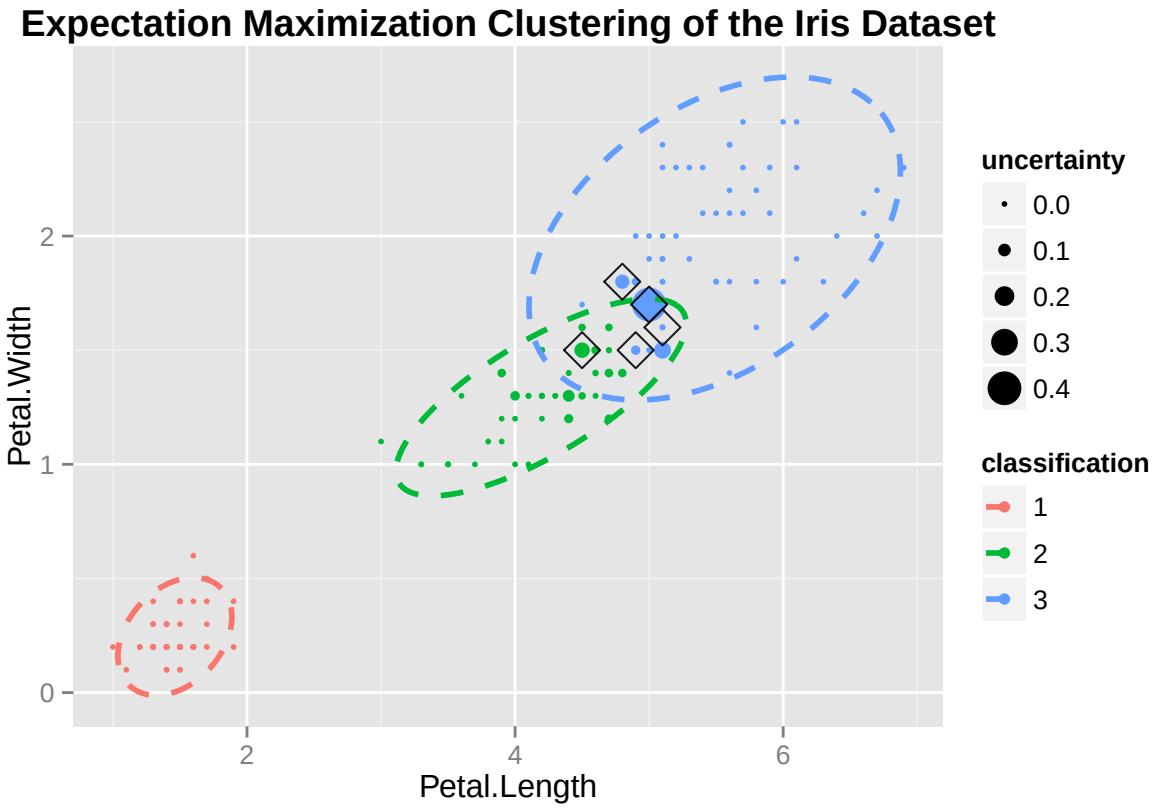
Visualizing and conceptualizing ICA is probably the most difficult algorithm of the whole lot. The algorithm finds the highest mutual information splits in the data sets and creates new features or dimensions from that analysis. It is very hard to visualize mutual information by just looking at the original dataset which is why it is hard to look at the transformed dataset and understand what the algorithm did to separate the data. When I visualize the output of the ICA algorithm, I simply try to understand if it was able to make separated clusters or not.

In essence PCA and ICA result in similar splitting of data through reducing the feature sets, but they do it in almost completely opposite ways. PCA is looking for k averages in the data, and clustering around that. ICA is looking for data that is different from other data, and separating clusters that way.

K-means Vs Expectation Maximization



In practice, K-means and Expectation Maximization provide very similar results. Below are the results of the Iris dataset when run through the algorithm. I could have chosen any of the four dimensions to look at, but for comparison I stuck to the first two. Below shows very similar clusterings. The main difference between EM and K-Means is that EM also returns an uncertainty variable for every data point that it classifies. This can go a long way in determining if your clustering algorithm is classifying data points arbitrarily or if it the algorithm returns good results with high certainty. For most of this project I decided to use EM algorithms whenever I could so that I could also look at uncertainty along with the misclassifications. To me, this made analysis of the different algorithms more interesting. It also helps that the EM algorithm also returns a better classification of the data being analyzed.

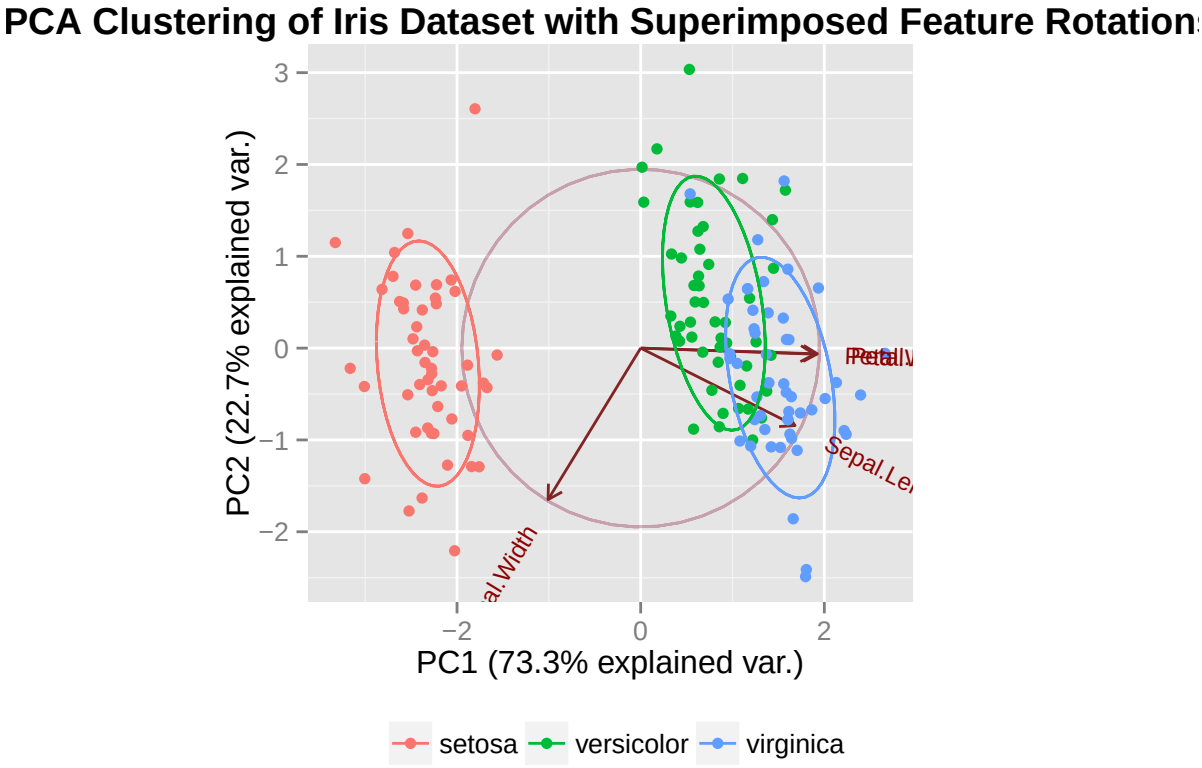


In the above graphs, and all others, the diamond around certain data points represents misclassified data. The Iris dataset provides an excellent example of a dataset where both EM and K-Means do a great job clustering all data points except for the data points near the boundary edge of two nearby clusters.

Principle Component Analysis

After creating a baseline of k-means and expectation maximization, the next step is to see if the features can be transformed in a way that allows for easier computation as well as extrapolation of important aspects of each of the features that results in a better classification. Below is a graph that shows the vectors of the original features (Petal.Width, Petal.Length, Sepal.Width, and Sepal.length) as they relate to the first two principle components.

This algorithm doesn't actually reduce the amount of features the Iris dataset has, but it certainly transforms the data. What is most noticeable with PCA is that two of the clusters overlap considerably more than the did when the dataset wasn't modified. Performance is similar to the expectation algorithm, but if anything, there is more uncertainty with PCA then there is with the original EM algorithm.



Independent Component Analysis

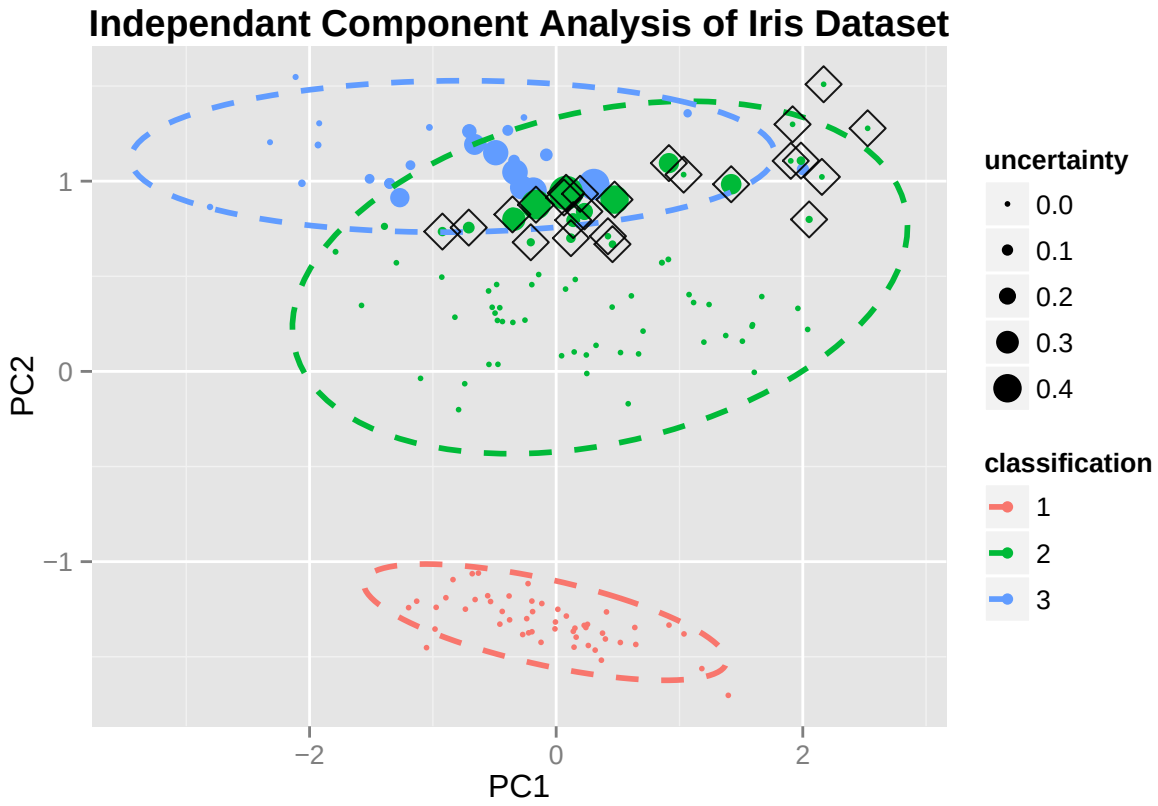
As stated earlier, independent component analysis is probably the most difficult of all of the algorithms to examine the result and understand what exactly the algorithm has done to the original features to replace them with transformed and possibly reduced ones. In theory, I believe I mostly understand how it would be possible to separate information based on mutual information, but it is very hard to look at a resultant graph of ICA and understand what the algorithm is actually separating. If I had prior knowledge of what some of the individual data really represented, I mat be able to gain some more insight.

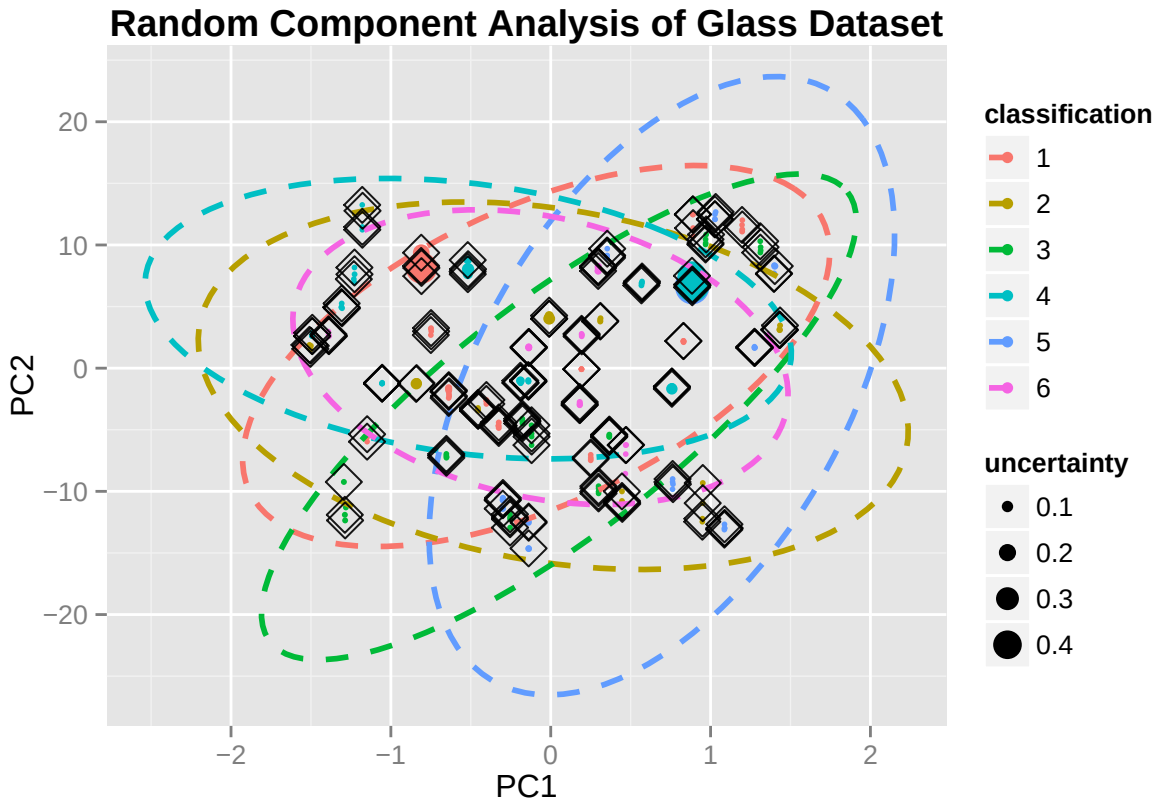
For the datasets I used, though, ICA generally performed worse than most of the others (except for some random projections). Despite this, I still believe that the algorithm has useful aspects that might readily apply to certain other datasets.

Random Component Analysis and Projections

The most simple way I found to implement random projections is to take the principal component analysis data structure, replace the rotation matrix it uses with a new random one, and rotate the data by the new matrix. Odds are that taking a random matrix and applying it with the data will create quite a mess, and result in interesting, but not necessarily useful clustering. The best illustration of this that I found was when I ran the Glass dataset with this algorithm (below).

There are many benefits to RCA, though. The first is that it can be much faster than some of the other algorithms, but the second is that, since it is a random projection you can use any of the random optimization algorithms to search for the best type of output. This is specifically useful in determining how many features are the right numbers of features to transform down to. The table below summarizes the performance of RCA when the projection matrix was randomly chosen 100 times. There were no optimization algorithms implemented with this, though it would be fairly easy to extend the analysis to do so.





	Max Error Percentage	Min Error PErcentage	Error Variance	Best Amount of Features
Glass Dataset	0.9346	0.6682	0.0017	2

	Max Error Percentage	Min Error PErcentage	Error Variance	Best Amount of Features
Iris Dataset	0.8133	0.3067	0.0103	3

Table 1: Iris and Glass RCA 100 Times

Information Gain Filtering

For my final algorithm, I chose to implement a filtering algorithm that would filter features by their initial information gain. For all of the datasets, I set the filter threshold to filter out any features that performed worse than the average information gain. For the Iris dataset, the algorithm removed two features. They were both of the features based on Sepal dimensions.

I thought that this might be a good algorithm to use, but I was surprised at just how good and simple it was. It didn't help much with the overall low performance of the Breast Cancer and Glass datasets, but it was the best performer of all of them for the Iris dataset.

Below shows the numeric values of the information gain of each of the Iris features, followed by a graph of all the features that were removed for the breast cancer dataset before they were analyzed.

	attr_importance
Sepal.Length	0.4521286
Sepal.Width	0.2672750
Petal.Length	0.9402853
Petal.Width	0.9554360

Table 2: Iris Importance

age	menopause	breast	breast.quad	irradiat

Table 3: Removed Features of Breast Cancer Data

Summary of Performance

Using the Iris dataset to become familiar with the clustering and feature transformation algorithms was a very good idea. I had originally picked the breast cancer and the glass dataset simply because they were very hard datasets to classify. Using the Iris dataset gave me a better glimpse as to why.

Categorical Breast Cancer

The breast cancer dataset can be remarkably hard to classify correctly. I believe there are multiple reasons for this, but one reason is certainly that the entire dataset is composed of categorical data. The iris dataset plugs into all of these algorithms very easily, but as soon as I tried to implement the breast cancer dataset, I found myself searching for good ways to measure distance and similarity of the breast cancer data. I initially broke the dataset up into binary strings, which actually made the amount of features multiply rapidly. Using

this type of dataset would be much easier to measure distance, but it also required me to separate all of the features into completely independent blocks. For example if two data points had a value of 40-45, and 45-50 for their age feature, the binary split would now have multiple features based on age. The first data point would have a 1 under age40-45, and the second would have a 1 under the age45-50 column. The data points would have a 0 in each of the rest of the age columns. Even though, there is clearly a relationship between different ages (even if there isn't a good distance metric to represent it), the binary dataset strips that relationship and considers every value to be completely independent of one another.

Even though the binary dataset removes a lot of the possible similarities from the data, it didn't seem to perform worse than any of the algorithms I had tested before this point. Eventually I decided that it was best to keep the data in its categorical form and only run algorithms that could handle that type of data, but I will elaborate on that in the Neural Net section.

Glass Data

The glass data is also a very hard dataset to pin down. Most of the graphs I found showed clusters that were tightly packed together. The algorithms were able to correctly classify some of he instances, but overall it performed very poorly. To illustrate my point, below is a confusion matrix table for the Glass data set after some of its features had been pruned. The confusion table illustrates some of the overall issues with being able to classify the data correctly

Summary of Performance Tables

Below I have three separate tables that highlight the overall performance of the algorithms for each of the three datasets. Overall, the information gain algorithm performed best on the Iris data set by having the least error. I was pretty proud of myself for arbitrarily picking an algorithm to test that ended up beating all the other algorithms.

Principal Component analysis performed the best on the "hard" datasets, which isn't surprising. In general, I don't think ICA or RCA had much of a chance to perform well on these datasets that had already proven to perform poorly on many other algorithms. Again, the Information Gain Filter also performed remarkably well considering the poor performance of the other algorithms.

I've included a cumulative uncertainty chart as well, as I found it useful to examine this metric as I was performing tests. In the case of ICA being run on the Iris dataset, there is much more uncertainty than all other algorithms, but the errors remain relatively low. This tells me that the algorithm got "lucky" in correctly guessing the algorithm and that new data might not easily fall into clusters that it has laid out. If given the choice, I would pick an algorithm that had both low errors and low cumulative uncertainty.

Finally, I've included a table of the percentage accuracy of the Glass and Breast Cancer datasets. This helps tie these algorithms to previous ones. The PCA and information gain algorithm were able to achieve success near the rates of original breast cancer training networks which is fairly remarkable given that there was no classification data used to create the clusters. The other datasets and algorithms were interesting to play around with, but did not return results that I would consider to be usable in application.

	Kmeans	EM	PCA	ICA	RCA	Information Gain Filter
Iris	16	16	9	24	105	4
Breast	NA	209	103	139	185	103
Glass	NA	160	142	147	181	159

Table 4: Cumulative Error

	Kmeans	EM	PCA	ICA	RCA	Information Gain Filter
Iris	NA	0.000000	3.2824231	7.009761	3.023822	4.6845906
Breast	NA	2.357744	2.4273209	1.672964	0.000000	0.6232290
Glass	NA	1.246969	0.0442991	3.303257	4.606315	0.8398738

Table 5: Cumulative Uncertainty

	Kmeans	EM	PCA	ICA	RCA	Information Gain Filter
Breast	NA	0.27	0.64	0.51	0.35	0.64
Glass	NA	0.25	0.34	0.31	0.15	0.26

Table 6: Percentage Accuracy of Breast Cancer and Glass

Analysis of Clustered Algorithms and Feature Transformed Data with Neural Networks

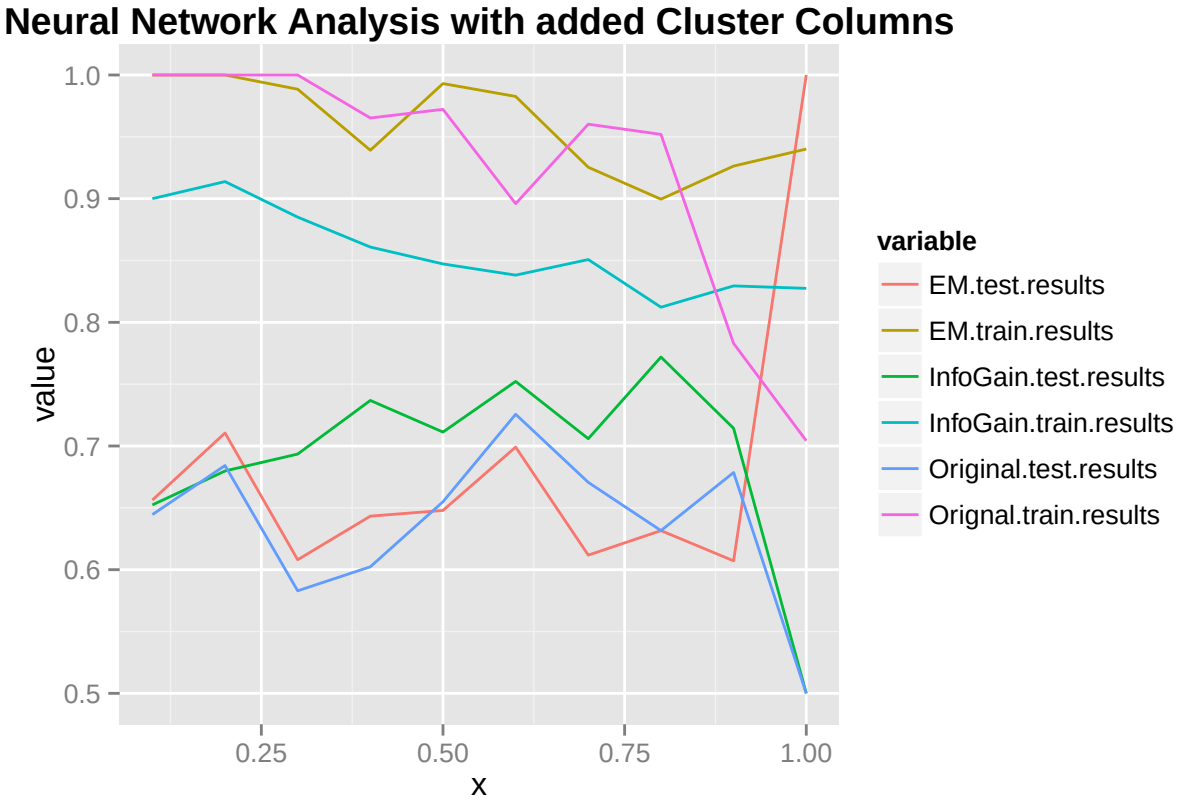
As I was shifting my attention to running neural networks again, I shifted the breast cancer data from a binary dataset back to categorical. I was unimpressed with the results, so I thought I might be able to find a way to implement better similarity metrics with the algorithms I was using. Unfortunately the R code I was using was a little too rigid to incorporate this kind of dataset. Luckily, I found a k-modes algorithm to replace the k-means algorithm. When I ran this new algorithm, I saw no change in performance. When I ran the EM algorithm, by itself, I saw the best performance that I had seen on all of the unsupervised algorithms. Below is a table that shows the raw performance of the categorical breast cancer data when run with the EM algorithm I was using.

Cumulative Error	83.00
Accuracy	0.71

Table 7: Categorical Breast Performace

Comparison to The First Assignment

Unfortunately, this improvement in unsupervised learning did not help the neural network perform any better. I compared a growing training set of both the EM and InfoGain clustered algorithms, and there is possibly a small improvement in the Info Gain test results, but it is nothing to write home about.



Summary

Unsupervised clustering and feature transformation algorithms are similar to many of the algorithms that we've seen before, but there are some very distinct differences. Performance of an unsupervised dataset seems to be much lower than the supervised dataset algorithms. When I think about this, it makes a lot of sense. If there isn't direct feedback in the algorithm, it will tend to perform worse. When searching on the Internet for algorithms and implementation techniques I quickly began to realize that these algorithms are best used with a prior knowledge of the data that you are working with. There is a possibility that you could get them to return excellent results by simply handing the data over to the computer, but it is much more likely that I would use these algorithms to illustrate properties of data that I suspected already existed.