Nathan Dass
CS 4641 – Assignment 3

Unsupervised Learning Report

Datasets

I used both of my datasets from the Supervised Learning assignment to observe the effect of various combinations of clustering (k-means and Expectation Maximization) and dimensionality reduction (Principle Component Analysis, Independent Component Analysis, Randomized Projections, and Information Gain) on the datasets. I also ran neural networks on the new datasets obtained from the dimensionality reduction algorithms and the dimensionality reduction algorithms combined with clustering algorithms.

Eye EEG. The first dataset I used tracked the state of an eye through an EEG. 14 EEG values were recorded and whether or not the eye was open or closed at a given time. The state of the eye was recorded through a camera and manually added to the data set by analyzing the video frames. The data was logged over the course of 117 seconds, giving about 128 measurements per second. The dataset had no missing values, contained 14 attributes that were all continuous, and 14980 instances.

Sick. The second dataset I used was a record of thyroid disease supplied by J. Ross Quinlan at the Garavan Institute from New South Wales Institute in Sydney, Australia. This dataset tries to determine whether or not a subject was sick or healthy (negative). This dataset contained a mix of continuous and discrete attributes. The dataset has 6064 missing values, which is 5.4% of all of the values. The dataset contained 30 attributes (7 continuous attributes and 23 discrete attributes) and 3772 instances. One thing to note is the high number of attributes in this dataset, but the lower number of instances. This plays into the Curse of Dimensionality, implying that we would actually need more instances to obtain accurate results.

Clustering

In this assignment, I used two clustering methods: k-means and expectation maximization. Below is a description of each of the algorithms.

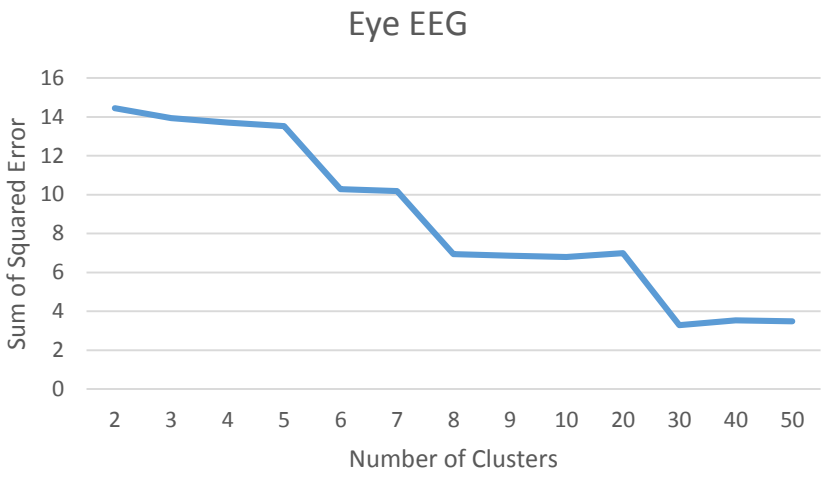
k-means: This algorithm assigns each point to the cluster with the nearest mean. K random points are chosen as a solution at first. The Euclidean distance between the data and the central points are calculated and the points are adjusted based on what the center of the clusters that are formed "claims." This process is repeated until a local maxima is achieved. To find the optimal value of k (number of clusters), I used the elbow method. I plotted the number of clusters as a function of the sum of squared error. Sum of squared error measures the discrepancy and differences between the data and the estimation model. From the graph, I found the point after which adding another cluster does not result in a significant change in the sum of squared error. This point of diminishing return is what I would use for my optimal number of clusters.

Expectation Maximization (EM): At a higher level, this algorithm assigns probabilities to each point that show how likely it is that this point is part of a cluster. This acknowledges the

fact that while we can almost guarantee that a point is part of a cluster, we are never 100% sure. With EM, I used log likelihood to determine how well similar data points are clustered together. The log likelihood indicates how well the data points fit to the cluster they are assigned to. I used Weka's cross validation for EM to average the log likelihood of a 10 fold cross validation to find the optimal number of clusters. Weka increases the number of clusters until the log likelihood stops increasing.

Eye EEG:

The graph to the right was used to determine a k of 8 for k-means with this dataset. I chose 8 because while the sum of squared error does decrease somewhere between 20 and 30, it is consistently around 6.8 from 8 until 20. Coupling this with the results obtained from expectation maximization resulted in the table shown to the right. Both algorithms determine a smaller number of attributes, but EM takes a significantly longer time to terminate. While EM is using cross validation, even if we divide the total time taken by a factor of 10 since there are 10 folds, EM still takes more than 20 times longer than k-means. This is probably because while EM never diverges, it never fully converges. EM practically always converges, but can also get stuck in a local maxima. Similar to randomized hill climbing and simulated annealing, random restarts can solve the local maxima problem, but require more time to reach a global maxima.



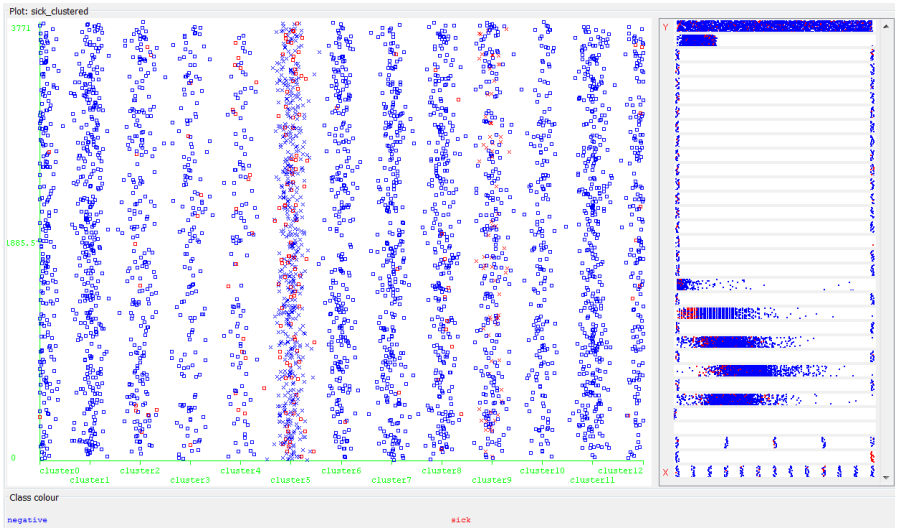
EM	
Clusters	5
Time (seconds)	254.79
Log Likelihood	-58.73449
k-means	
Clusters	8
Time (seconds)	1.24

Sick:

EM	
Clusters	13
Time (seconds)	137.34
Log Likelihood	-5.8765
k-means	
Clusters	5
Time (seconds)	0.3

The results of the Sick dataset were similar to the results of the Eye EEG dataset. EM still took significantly longer than k-means. In this case, however, EM resulted in a much higher number of clusters than k-means. However, it should also be noted that the log likelihood is very close to 0, implying that EM worked well for this data.

Why is EM splitting this dataset into so many clusters? Through Weka's visualization tools, I was able to see a distribution of the clusters as shown below. The x axis are the respective clusters while the y axis is instance number. It was interesting that within each of the clusters, there is a relatively high amount of instances in each cluster. One cluster did not contain a large majority of the dataset. I think this could be due to the fact that this dataset falls into the Curse of Dimensionality since the number of attributes is not very high compared to the number of features.




```
Clustered Instances

0       227 (  6%)
1       380 ( 10%)
2       239 (  6%)
3       167 (  4%)
4       129 (  3%)
5       625 ( 17%)
6       279 (  7%)
7       321 (  9%)
8       319 (  8%)
9       315 (  8%)
10      199 (  5%)
11      387 ( 10%)
12      185 (  5%)
```

Dimensionality reduction with and without clustering

Dimensionality reduction reduces the number of attributes in a dataset in order to better model it. There are multiple ways of going about this process. In this assignment, I used four dimensionality reduction algorithms: principle component analysis, independent component analysis, randomized projections, and information gain. These algorithms where then combined with the clustering algorithms previously used.

Principle component analysis (PCA): PCA takes a linear combination of the original features to make new features and ranks them from highest variance to lowest variance across the dimension in directions that are mutually orthogonal. PCA strives to maximize variance simply because a higher variance in a dataset will lead a wider scope.

Eigenvalue	Proportion	Cumulative
4.09447	0.29246	0.29246
3.34629	0.23902	0.53148
2.74196	0.19585	0.72734
2.62971	0.18784	0.91517
0.49576	0.03541	0.95059

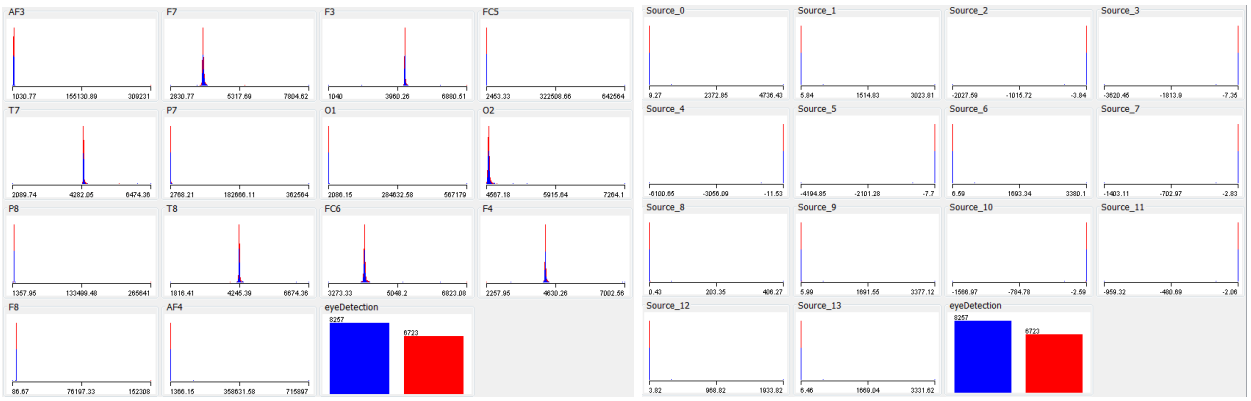
Eigenvalues are an integral part of PCA. The eigenvalues are guaranteed to monotonically non-increase as you go up in dimensions. They also play a role in variance in the sense that if an eigenvalue is equal or very close to 0, that dimension does not provide any new information and can be removed from the dataset. For example, the table above shows the eigenvalues obtained when using a variance of 0.95 on PCA. The last eigenvalue is pretty close to 0 and really does

not contribute much to the dataset. Therefore, we can just throw out the dimension associated with the last eigenvalue.

Dataset	Clustering Algorithm	Variance Covered	Number of attributes	Log likelihood	Time elapsed (seconds)	Number of Clusters
Sick	k-means	0.25	3		0.17	10
Sick	k-means	0.5	8		0.34	25
Sick	k-means	0.75	16		0.5	35
Sick	EM	0.25	3	-3.61537	878.48	30
Sick	EM	0.5	8	-6.70522	690.93	24
Sick	EM	0.75	16	-9.91587	375.17	16
Eye EEG	k-means	0.25	1		0.17	2
Eye EEG	k-means	0.5	2		0.14	4
Eye EEG	k-means	0.75	3		0.18	4
Eye EEG	EM	0.25	1	-0.58844	54.5	2
Eye EEG	EM	0.5	2	-0.26545	29.48	3
Eye EEG	EM	0.75	3	-0.81586	81.27	2

I changed the variance covered of the PCA algorithm because this related to the proportion of variance that must be retained by the different principle component attributes. I found a few interesting things when applying the PCA filter to both datasets. First, the same trend followed where EM took significantly longer to run than k-means, almost taking 15 minutes for a single run. Second, even though the eye dataset has close to 15,000 attributes and 14 attributes while the sick only has 3772 attributes and 30 instances, EM took about 10 times longer for the sick dataset even though it had fewer attributes. Third, the sick dataset resulted in a larger number of clusters, even though the elbow method used with k-means. Fourth, the log likelihood of all of the results for EM are all very close to 0, indicating that EM works pretty well with PCA.

Independent component analysis (ICA): ICA maximizes the independence of the transformed feature space. It ensures that new features are independent of each other and that the information between original features and transformed features is maximized. ICA has this underlying assumption that we can see the observable variables in the world and we want to use these observable variables to find the hidden variables that we don't know about. The key difference between PCA and ICA is that ICA tends to find parts of the whole while PCA is more of a global algorithm and is forced to find global features.



ICA also assumes that things are not only independent, but also highly non-normally distributed. This is where kurtosis comes in. Kurtosis is the measure of tailedness and non-normality and since this is exactly what ICA is aimed to achieve, ICA tends give datasets that have a high kurtosis. This is because a higher measure of kurtosis means the distribution is more focused on the peaks, meaning the other attributes do not have as much of an influence on the current attribute. By looking at the distributions above of the attributes before applying ICA (left) and after applying ICA (right), I could easily see the distributions became more focused on the peak, so the kurtosis definitely increased.

Dataset	Clustering Algorithm	Tolerance	Number of attributes	Log likelihood	Time elapsed (seconds)	Number of Clusters
Sick	k-means	1.00E-02	33		1.07	35
Sick	k-means	1.00E-04	33		0.92	40
Sick	k-means	1.00E-06	33		0.42	30
Sick	EM	1.00E-02	33	118.6583	2989.68	34
Sick	EM	1.00E-04	33	106.467	546.75	13
Sick	EM	1.00E-06	33	108.1284	545.31	12
Eye EEG	k-means	1.00E-02	14		0.12	3
Eye EEG	k-means	1.00E-04	14		0.07	3
Eye EEG	k-means	1.00E-06	14		0.44	4
Eye EEG	EM	1.00E-02	14	-46.0292	5.78	2
Eye EEG	EM	1.00E-04	14	-61.4132	5.97	2
Eye EEG	EM	1.00E-06	14	-51.6577	7.74	2

I decided to change the tolerance when testing ICA because the tolerance signified the error tolerance for solution convergence and led to how soon or late the algorithm would converge. A lot of the trends observed in the results of PCA were also seen here as well. The sick dataset had many more clusters, EM took longer than k-means, and the sick dataset took longer than the eye dataset. However, the log likelihood is very different from the PCA results. The log likelihoods for the eye dataset are all larger negative numbers while the log likelihood for the sick dataset are all large positive numbers. This would imply that EM worked well for the sick

dataset, but did not work well for the eye dataset. Since the results were either consistently bad or consistently good depending on the dataset, I do not think the clustering algorithm is the problem. Instead, I think it is the ICA algorithm itself that is causing these log likelihood. As mentioned earlier, ICA tries to maximize independence. However one big drawback is that sometimes, you cannot find independent components for a dataset. I decided to look further into this trend and thought about the attributes of the datasets. The sick dataset contains a variety of attributes, including attributes that are completely unrelated such as age, sex, pregnant, thyroid surgery. Between these four attributes themselves, it is hard to see any dependencies. On the other hand, the eye dataset contains data coming from a single EEG device. Since the data is coming from the same source, some of the data is bound to be dependent on each other. This would explain why ICA does pretty badly on the eye dataset, but works well with PCA. ICA only works on datasets that are discretized and do not have missing values, so I was originally unable to apply ICA to the sick dataset. I used the NominalToBinary and ReplaceMissingValues filters for the discrete attributes and missing values.

Randomized projections: Randomized projections takes the current data and generates random directions to project the data onto. Even though this algorithm is arbitrarily reducing the dimensions, it still maintains some of the correlations in the dataset.

Dataset	Clustering Algorithm	Seed	Number of attributes	Log likelihood	Time elapsed (seconds)	Number of Clusters
Sick	k-means	25	10		0.98	40
Sick	k-means	50	15		0.47	20
Sick	k-means	75	20		0.56	20
Sick	EM	25	10	-37.1061	36.16	6
Sick	EM	50	15	-63.1486	63.76	6
Sick	EM	75	20	-39.8028	1723.22	31
Eye EEG	k-means	25	10		0.29	3
Eye EEG	k-means	50	15		0.22	6
Eye EEG	k-means	75	20		0.34	5
Eye EEG	EM	25	10	-54.9782	56.69	3
Eye EEG	EM	50	15	-82.5081	106.89	3
Eye EEG	EM	75	20	-109.152	192.48	3

Since randomized projection starts off by generating the random matrix, the way that this matrix is created plays an important role in the outcome of the algorithm. That is why I decided to change the seed parameter because this is used by the random number generator when the random matrix is being generated. What I found particularly strange about the using randomized projections was the nature of the results themselves. There was simply no clear trend in the data obtained. The log likelihood jumped all over the place, the times were not consistent, and the number of clusters were consistent for the eye dataset but varied significantly for the sick dataset. I think these erratic results can be explained by the nature of the algorithm. Since it does rely

heavily on random chance, the choice of the seed can place the algorithm in a position to end quickly or put the algorithm in a very bad position that will get stuck. I think this goes to show that randomized projections is not a very reliable algorithm and there are other dimensionality reduction algorithms that are faster and give better results.

Information Gain: Similar to the process used for decision trees in the Supervised Learning assignment, this algorithm ranks the attributes based on how much unique information they contribute to the dataset as a whole. After this list is obtained, it is easy to remove the attributes that are not very helpful or original.

Dataset	Clustering Algorithm	Attributes Removed	Number of attributes	Log likelihood	Time elapsed (seconds)	Number of Clusters
Sick	k-means	19	10		0.49	45
Sick	k-means	14	15		0.84	45
Sick	k-means	9	20		1.02	45
Sick	EM	19	10	-9.35409	20.11	6
Sick	EM	14	15	-11.3348	43.72	9
Sick	EM	9	20	-12.2993	48.66	9
Eye EEG	k-means	9	5		0.26	8
Eye EEG	k-means	6	8		0.92	10
Eye EEG	k-means	3	11		0.83	10
Eye EEG	EM	9	5	-23.7981	29.98	3
Eye EEG	EM	6	8	-49.9306	38.3	2
Eye EEG	EM	3	11	-45.9738	164.54	2

Sick	
0.173159	T3
0.0.060069	referral source
0.041154	T4U
0.022877	age
0.01622	T3 measured
0.013921	TT4
0.006656	TSH measured
…	…
2E-06	lithium
6.75E-07	goitre
0	TBG measured
0	TBG
0	FTI
0	TSH

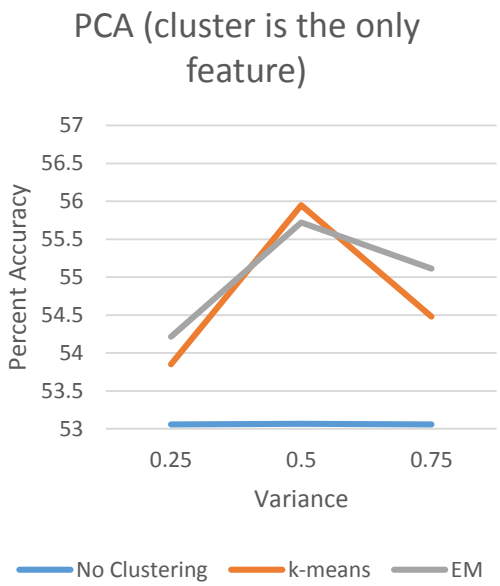
Eye	
0.0598	O1
0.0572	P7
0.0493	AF3
0.046	F8
0.0323	F8
0.0257	F4
0.0223	P8
0.0216	T8
0.0194	FC6
0.0174	T7
0.015	O2
0.013	FC5
0.0129	F7
0.0124	F3

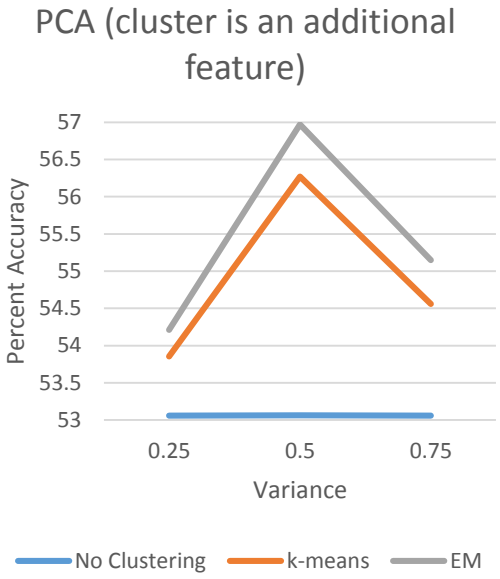
By glancing at the results obtained after combining Information Gain with k-means and EM, I noticed that the times were actually higher for the eye dataset than the sick dataset and the log likelihood was also significantly worse for the eye dataset than the sick dataset. Also, this was the first time that I saw the number of clusters being close to 10 for the eye dataset, which was obtained using k-means. By looking into the actual ranking that the Information Gain provides for each of the data sets, it is easy to see why the algorithm did well for the sick dataset, but did poorly for the eye dataset. None of the attributes really provide a large amount of information about the dataset. They all

provide a pretty low amount, so they are all mostly necessary for the clustering algorithm to do well. The seed would determine the random starting and affect the resulting dataset, leading to results that are not reliable or stable. Part of the information gain table is shown for the sick dataset as well. As opposed to the information gained by the eye dataset, a few of the attributes contribute a large portion of information and a lot of the attributes have a contribution close to 0. When these attributes are removed, they do not have a big impact on the resulting dataset itself and that is why the results for the sick dataset do not vary as much depending on the seed.

Neural network comparisons

In order to see the actual effectiveness of the different combinations of the clustering and dimensionality reduction algorithms, I ran the new datasets obtained in a neural network. I used the standard learning rate of 0.3, momentum of 0.2, and training time of 500 when running the neural networks. I modified the new dataset and ran them in a neural network in two different ways. In the first way, I treated the new "cluster" attribute as an additional attribute to the rest of the attributes and in the second way, I treaded the new "cluster" attribute as the only attribute and removed the rest of the attributes from the dataset. For the most part, there was not much of a difference in the percent accuracy between the two methods. While ICA saw about a 2% increase in one of the EM trials, it was really the PCA values that were amplified by treating the "cluster" attribute as an additional attribute. However, percent accuracy is not the only thing to try to improve. It is also important to look at time taken to build the neural network itself since it is not worth it to have an extra percent of accuracy at the cost of doubling the build time.





To compare the data, I used some of the data collected from the Supervised Learning assignment as a baseline for the percent accuracy and build time for a neural network with the same complexity. I determined the build time and the percent accuracy of the different algorithms independently so their direct values can be compared to the values obtained by the original dataset.

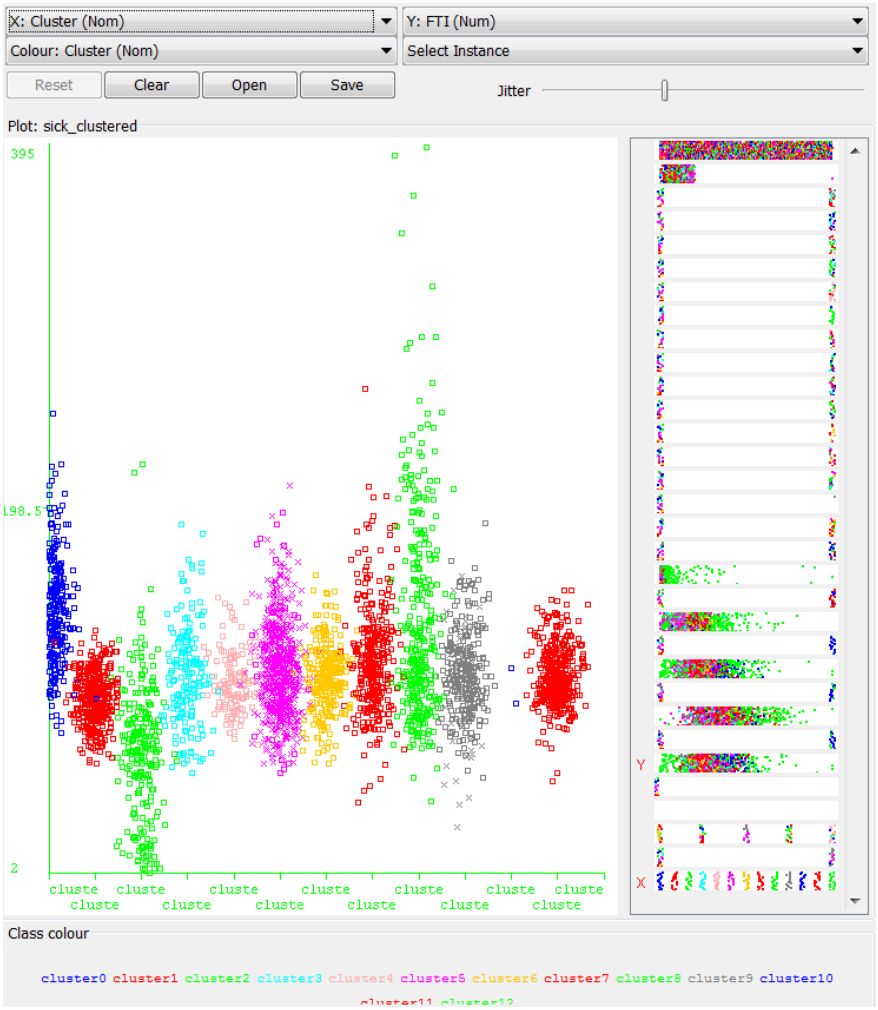
	Time (seconds)	Percent Accuracy
Original	22.83	54.8064
Average for all modified datasets	19.437	55.3869
PCA	14.758	54.63419
ICA	28.834	56.45636
Randomized Projections	15.404	54.1651
Information Gain	18.752	56.29193
No Clustering	11.94	53.53638
k-means (solo)	14.86	55.66366
EM (solo)	12.73	55.72343
k-means (added)	29.8775	55.78438
EM (added)	27.775	56.22663

As a whole, it looks like modifying the original dataset either through dimensionality reduction or dimensionality reduction and clustering decreases the build time and increases the percent accuracy. Going into more specifics, when the clustering algorithm was treated as an additional attribute, the build time increased significantly, but it also yielded some of the higher percent accuracies. However, information gain performed the best in terms of percent accuracy and was also had a fast build time. What did surprise me was the build time of EM. During the results gained in earlier parts, EM was always the slowest algorithm, especially when combined with KNN. Regardless, applying any of the algorithms used in this class can and should be executed since there is no harm caused by anything that might come out of it. Randomized projections had one of the fastest build times, but also had the lowest accuracy. I think that this supports that randomized projections is just not a very reliable algorithm when compared to PCA or ICA or Information Gain. Seeing that ICA had the highest percent accuracy also supports my previous hypothesis that the Eye EEG dataset had many features that were naturally independent of each other, so it was not that hard for the algorithm to perform well.

Conclusion

From this assignment, I learned about different unsupervised learning algorithms, specifically for clustering and dimensionality reduction. I learned about the pros and cons of k-means and expectation maximization for clustering and principle component analysis, independent component analysis, randomized projections, and information gain for dimensionality reduction. With what I learned, I was able to compare the effects of these algorithms and revisit supervised learning to see how unsupervised learning techniques can help with labeling data, function approximation, and reducing overfitting and tackling the Curse of Dimensionality by generalizing the dataset.

I also found the sum of squared error that I got for the eye and sick datasets to be curious. To put it in perspective, the average sum of squared error for the eye dataset was 11.2205 while the average sum of squared error for the sick dataset was 3820.0717. I think this might be due to the nature of the attributes of each dataset. The eye dataset has all continuous values, while the sick dataset has a mix of continuous and discrete values. Similar to what was established in the Supervised Learning assignment with regards to kNN, continuous data often portrays how locality is actually meaningful. In other words, values of attributes that are close to each other bear some actual significance. Of course, this all depends on the actual dataset itself and what the attributes represent. In the case of the eye dataset, since all values are being measured, collected, and recorded it would make sense that recorded values that are close to other values share some resemblance in the underlying distribution that they represent and whether or not the eye is actually closed or open. On the other hand, discrete datasets (like the sick dataset) do not necessarily follow this trend of having locally meaningful values.



I learned a lot about the interaction between different algorithms and how to analyze the interactions in qualitative and quantitative ways through kurtosis, elbow method, visualization. This assignment gave me an in depth exposure and really emphasized the importance of optimality through finding the optimal number of clusters for each combination of dimensional reduction and clustering. I also learn about how combining different algorithms can either help or harm the overall performance and speed. It is important to keep in mind what each algorithm is geared towards, what the underlying assumptions are, and which cases the algorithm can excel at. Finally, the characteristics of the actual dataset itself should be taken into account. No two datasets are the same and each dataset has unique qualities that certain algorithms can be tailored to capture and highlight.