UNSUPERVISED LEARNING & DIMENSIONALITY REDUCTION

David Gong
Department of Computer Science
Georgia Institute of Technology, davidcgong@gatech.edu

March 25, 2019

ABSTRACT

This report seeks to explore unsupervised learning algorithms (clustering and dimensionality reduction) using WEKA. These algorithms include k-means clustering, Expectation Maximization, PCA, ICA, Randomized Projections, and Information Gain. Additional neural network learner tests were also conducted on reduced data.

1 Datasets

The datasets used to conduct Unsupervised Learning and Dimensionality Reduction analysis do not change from the ones used to conduct Supervised Learning analysis. The two datasets used are from the UCI Machine Learning Repository - Spambase and Poker Hands.

The Spambase dataset contains 4601 instances of data and 56 predictive attributes, many of which vary significantly in terms of feature importance and should be kurtotic in nature. The Poker Hand dataset contains 25008 instances of data and 10 predictive attributes, and follows the Poker probability distribution with discrete values. This is distributed and skewed in the sense that higher-likelihood hands are frequented much more often in the data, though the values are discrete.

As a result, prior to testing the algorithms and extracting the results, it can first be predicted that the dimensionality reduction algorithms should have at least moderate implications for our Spambase dataset. Our Poker Hand dataset will also likely have implications with clustering algorithms, though dimensionality reduction will not have as much of an effect on the dataset.

2 Clustering (k-means & Expected Maximization)

2.1 Spambase

An initial exploration with non-reduced Spambase data was conducted with k-means and Expectation Maximization clustering. The default settings were used, where k was set equal to the number of possible nominal class attribute values (spam or non-spam) and Euclidean distance was used over other distance functions for both clustering algorithms.

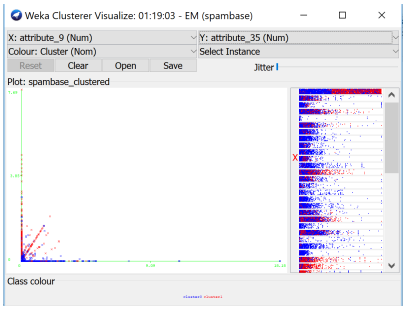
2.1.1 k-means

Using the default settings, k-means clustering incorrectly clustered instances 20.0826% of the time ($\frac{924}{4861}$), and took 0.2 seconds to train. This is a relatively interesting result, as it would typically be expected that having a large number of features would result in difficulty in clustering. The features were also thought to be generally stochastic and uniform in weight when it comes to spam identification.

As will be revealed by Information Gain, it was found that when attempting to use a sparse matrices approach to filtering out classes with less instances, there were a few features that were more important in general than others. This could be a potential reasonable explanation for the ability for k-means clustering to be able to cluster better than expected. As also can be seen from the graph, a clear line could be drawn to distinguish the clusters for most two features compared.

2.1.2 Expectation Maximization

EM incorrectly clustered instances 22.8429% of the time ($\frac{1041}{4601}$), and took 0.61 seconds to train. The accuracy is slightly less than that of normal k-means clustering and took somewhat longer to train.



As can be seen in the relative graph, when the two features (feature 9 and feature 35) combinations are measured, a distinction by line between the clusters is visible. On a higher level, there seemed to be significantly less overlapping between data however. Many cluster1 and cluster0 data points do not intersect as much as they do with normal k-mean clustering.

This is due to the nature of the EM algorithm. The EM algorithm "guesses" cluster centers and repeats until convergence, where the expectation step assigns points to the nearest cluster center and the maximization step sets cluster centers to the mean. When this happens, the algorithm is more likely to make discerning distinctions than that of normal k-means clustering.

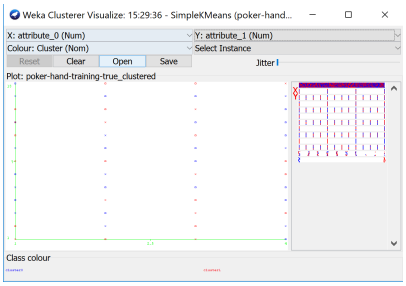
2.2 Poker hand

2.2.1 k-means

Using the default settings, k-means clustering incorrectly clustered instances 53.1427% of the time ($\frac{13291}{25010}$) and took 0.1 seconds to train.

Unsurprisingly, the k-means clustering algorithm had difficulty in properly clustering the poker hand dataset due to the unique probability distribution of poker. As mentioned in the previous supervised learning paper, the data points are heavily skewed towards 0 and 1 (one-pair and no-pair values) and then drop heavily from there. As a result, creating 2 clusters was tested to be optimal, which still resulted in low correctness observed.

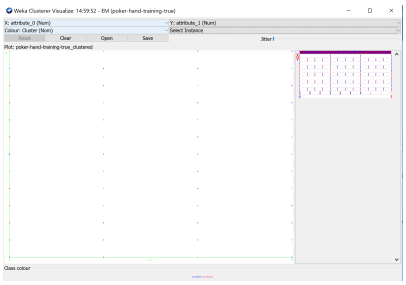
Attempting to create 10 clusters led to an over 88% inaccuracy rate. The identified one-pair and no-pair values would be separated into the different clusters rather than being grouped in the same cluster, which would ultimately drastically reduce accuracy.



The data visualization also shows that there is no clear distinction between which cluster is which and there is a lot of overlapping. These overlapping values lead the algorithm to incorrectly cluster many instances, resulting in the sub-50% clustering accuracy.

2.2.2 Expectation Maximization

EM incorrectly clustered instances 53.4066% of the time ($\frac{13357}{25010}$) and took 0.98 seconds to train. In this case, EM performed identically in terms of performance, which is also backed accordingly to the striking similarity between the normal k-means data visualization.



The likely reason that EM performed similarly to k-means clustering in this case is because with discrete values for every single predictive attribute, EM more or less adopts a similar numerical approach in terms of calculating the centroid for clusters. A discrete dataset would be much better at figuring out the means and clusters of a data set in general, but the problem with this dataset largely lies within the inherent probability of poker. The data is heavily screwed to the left towards no-pairs and one-pair values As a result, neither of these approaches proved to be fruitful in generating meaningful results.

3 Dimensionality Reduction

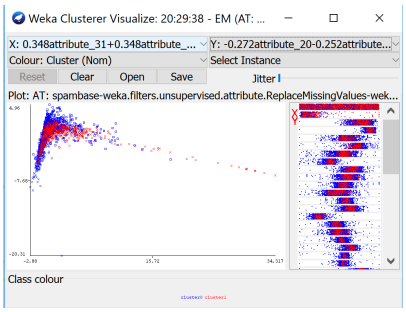
Four approaches were used for this section (PCA, ICA, Randomized Projections, and Information Gain). Half the number of attributes were reduced for each approach, with 0.95 variance. A few revelations prior to analyzing independent algorithms are as follows:

- Re-running the DR algorithms lead to minor differences in cluster generation, but remain the same overall, unless variables such as variance, and of attributes were changed.
- Higher variance in general leads to much more accurate results, though it takes longer to train the data. As an example, a test with 0.25 variance leads to generally training times that are multiplicatively shorter than that of tests with 0.95 variance.

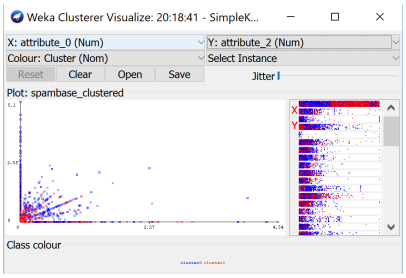
3.1 PCA

3.1.1 Spambase

The first eigenvalue produced had a value of over 6, the second had a value of over 3, the third had a value of 2, and the resulting eigenvalues decayed in value linearly. We then cut the number of features by half after performing the data transformation to get the following results:



k-means clustering: Incorrectly clustered instances: 20.7564%, Training time: 0.05 seconds



EM: Incorrectly clustered instances: 42.2734%, Training time: 1.08 seconds

The k-means clustering performed as expected, no significant hits to accuracy and took twice as fast to train, which is the entire goal of dimensionality reduction.

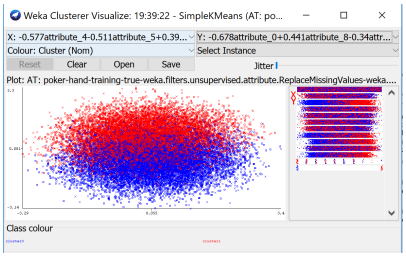
An interesting note worthy of observation here is the severe reduction in accuracy for our EM algorithm. In addition, it even took nearly twice as long to train as did EM before dimensionality reduction. A plausible explanation for this is that dimensionality reduction reduced the effectiveness of the distance function that previous attributes had. As a result, the calculation for finding the probability and placing the data point in the most likely cluster took longer to compute.

3.1.2 Poker hand

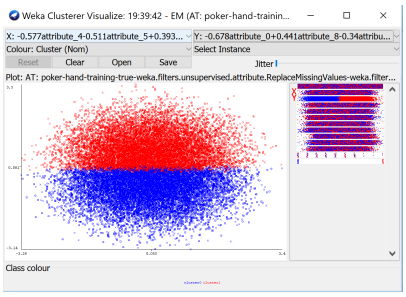
The eigenvalues remained relatively the same, ranging from 0.9 to 1.1. This seems to suggest that relatively speaking, the information that the attributes provide are all relatively important to the dataset.

A point to mention prior to conduction of dimensionality reduction on the dataset is that to identify poker hands, you need all the predictive attributes in order to generate a sure decision about what kind of poker hand has been drawn. That said, with the goal of generalizing the data and due to poker's skewed distribution, the goal of dimensionality reduction in this case is to be able to emphasize additional feature importance with regards to the first few cards drawn and to increase overall performance.

Here were the following results after applying k-means and EM to the PCA dataset:



k-means clustering: Incorrectly clustered instances: 53.7985%, Training time: 0.16 seconds



EM: Incorrectly clustered instances: 53.1667%, Training time: 4.36 seconds

In contrast to the previous example, when the dimensionality reduction algorithm was applied, there became a much clearer distinction between the two clusters within the data, especially with EM. With a normal probability distribution across the data and reduced features, EM became much more adept at identifying numerical distinctions between the clusters as there were less means to analyze and normalize. Essentially, the probability of poker was converted to a probability similar to that of a Gaussian one after PCA was applied. In addition, the clustering distinction was done largely in due to low positive kurtosis.

An interesting observation to note once again is the quicker time-performance of normal k-means clustering, but once again the slower time-performance for EM. This can likely be attributed once again to the increasing cost due to the decreasing effectiveness of attributes for computing the distance function.

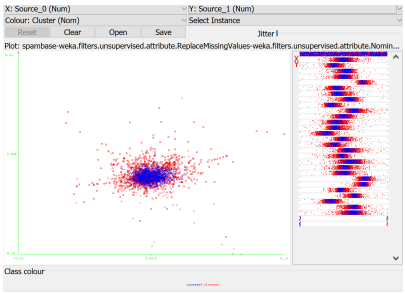
The algorithms still do not perform very well in terms of accuracy when it comes to correctly clustering instances, but that is expected as accuracy metrics remained relatively similar prior to conducting dimensionality reduction.

Overall, the goal of dimensionality reduction with PCA is achieved for both datasets with normal k-means clustering, but not for EM.

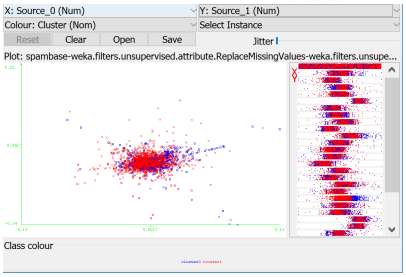
3.2 ICA

Independent Component Analysis is used for revealing and individually identifying hidden factors for different sets of random variables and measurements. It should be expected that in contrast to PCA, the clusters would be harder to make out and that there should be overlapping between the data for different clusters. ICA would excel at making out specific instances in noisy data.

3.2.1 Spambase



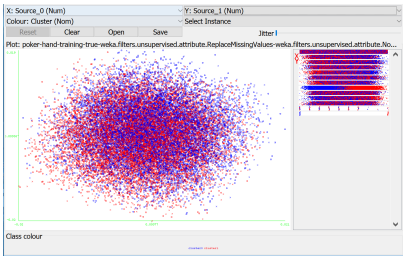
k-means clustering: Incorrectly clustered instances: 41.7953%, Training time: 0.11 seconds



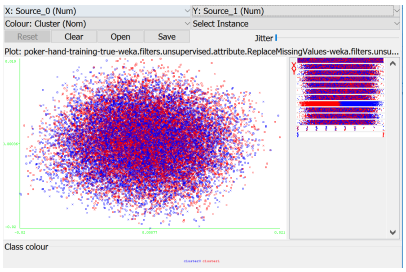
EM: Incorrectly clustered instances: 49.4458%, Training times: 1.74 seconds

With ICA, it appears that the centroids ended up being at the wrong areas of the graph. This can likely be attributed to the fact that having with the large amount of relevant attributes, ICA seemed to be unable to provide proper generalizations to the data. This results from high kurtosis, which was made evident when viewing the probability distribution of the dimensionally reduced dataset. As a result, accuracy was reduced, though training time was also marginally reduced as expected due to the nature of dimensionality reduction

3.2.2 Poker hand



k-means clustering: Incorrectly clustered instances: 53.6835%, Training time: 0.19 seconds



EM: Incorrectly clustered instances: 53.4306%, Training times: 2.93 seconds

In this case, the accuracy still remained relatively the same and there were slight time-performance improvements for EM. The data visualizations, however, showed a different story. There seemed to be a lot of overlapping of data between the clusters, which did not result in a clear distinction between the two clusters, even with the use of EM. The plausible explanation can most likely be attributed to the attributes of the poker dataset, because the predictive attributes are all needed in order to make a definitive conclusion about the classification of the poker hand.

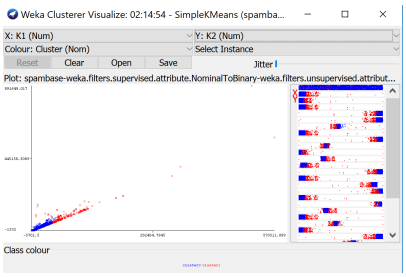
3.3 Randomized Projection

Randomized Projection (RP) conducts dimensionality reduction by projecting onto a lower dimensional subspace using a random matrix with columns of unit length. Similar to PCA in that sense, but the computational cost is also significantly reduced.

As a preface prior to running RP, the best case scenario would be that the time-complexity will decrease a lot compared to previous iterations of PCA and the overall accuracy is found to be similar. The reduced data have the same number of attributes as usual (halved) and have seed 42. Also, since this is randomized projection and that there will ultimately be some variance for each time the filter is applied, the average of 6 runs were used for Incorrectly clustered instances and the longest time was chosen.

3.3.1 Spambase

k-means clustering: Incorrectly clustered instances: 36.43%, Training time: 0.03 seconds



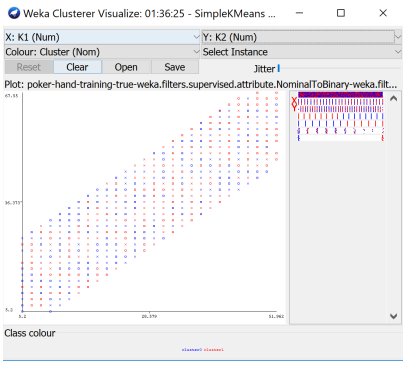
EM: Incorrectly clustered instances: 30.33%, Training time: 0.6 seconds

Every run iteration maintained relatively the same amount of time, and maintained inaccuracy differences of plus/minus 5%. The elbow method visual also supports the theory that in terms of clustering, randomized projection did relatively well. As a result, the results can deemed to be reliable overall. The clusters can clearly be distincted based on the graph, though EM seems to have been able to accurately differentiate between no-pair and one-pair values than normal k-means clustering after applying the RP filter.

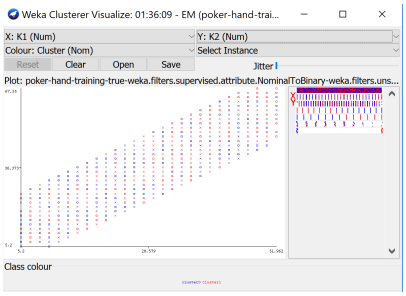
A likely explanation for this is simply due to the change of means and numerical values after conducting RP. These values and the adoption of a sparse matrix allowed the distance calculations to be quicker and also more accurate.

However, the same cannot be said for normal k-means clustering which seemed to have trouble here. This could be quickly attributed to the high kurtosis of the overall data each time it was transformed through RP, which made it harder to construct correct cluster centroids.

3.3.2 Poker hand



k-means clustering: Incorrectly clustered instances: 53.6505%, Training time: 0.12 seconds



EM: Incorrectly clustered instances: 53.56%, Training time: 0.77 seconds

For this case as well, every run iteration maintained relatively the same amount of time and inaccuracies, so the results should be generally reliable.

As can be seen, the Randomized Projection dimensionality reduction on the data did not seem to impact k-means clustering nor EM by much at all (inaccuracy stayed within 53 and 54 range), but there was a time improvement by a factor of 6 when it came to applying EM after RP as compared to PCA. As a result, RP did perform as we expected, and

this can largely be attributed to how RP project onto a subspace, which only takes O(nks) time where s is subspace size and matrix is n x k size. This, in addition to utilizing a sparse matrix led to drastic performance improvements.

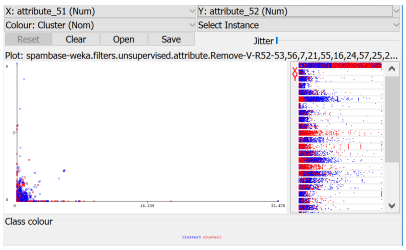
PCA in general, however, should be more accurate than that of RP. In this case, this wasn't the case once again due to the skewing of data and tendency to identify no-pairs and one-pairs over any other results. One key thing to note, however, is how RP did not seem to be able to reduce the previous kurtosis values for our previous data and that the distribution did not adhere as perfectly as PCA did.

As a result, there could possibly be a strong correlation between the data and the results for this specific case, instead of dimensionality reduction methodology.

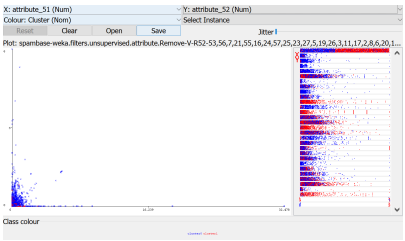
3.4 Information Gain

The goal is to use information gain feature selection algorithm to reduce a bias towards multi-valued attributes by choosing an attribute through considering number and size of decision tree branches. Through using information gain, the first half ranked attributes were used for transforming and reducing the data.

3.4.1 Spambase



k-means clustering: Incorrectly clustered instances: 20.235%, Training time: 0.03 seconds



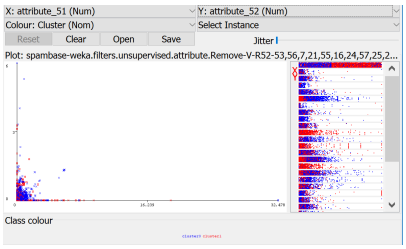
EM:
Incorrectly clustered instances: 13.432%, Training time: 0.39 seconds

In this case, information gain was the best feature selection algorithm out of all the previous ones. This is primarily because of how information gain evaluates the worth of an attribute by measuring the information gain with respect to the class.

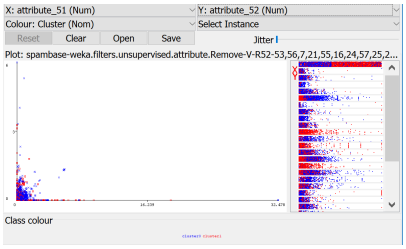
The k-means clustering with the information gain reduced dataset worked as well in terms of accuracy and took faster to train than normal k-means clustering, but EM with the dataset actually improved the accuracy of the dataset, along with the training time. A probable explanation for this phenomenon is that with the reduced number of attributes in the dataset, there became a reduction noise and jitter within the data. Reducing the number of attributes and selecting the most relevant ones in accordance with the ranking of information gain is what I believe ultimately allowed EM to make predictions based on less but more accurate numerical values.

3.4.2 Poker hand

Before conducting any analysis, information gain seemed to be able to reach the conclusion that all attributes are equally ranked (attributes all ranked 0), which is supportive of the idea that one probably needs to see all the cards within a hand to make a decision/conclusion about which poker hand is being shown. Regardless, half the attributes were cut to see if there were any changes that occurred.



k-means clustering: Incorrectly clustered instances: 53.1427%, Training time: 0.04 seconds



EM: Incorrectly clustered instances: 53.5506%, Training time: 1.04 seconds

As can be observed, there were no noticeable changes in the accuracy as compared to running clustering on non-reduced data, though the performance did improve from reducing the number of attributes that had to be taken into account. This is simply because in information gain for this case, the attributes are all ranked the same relative to the class. If this was not a poker probability distribution that was so skewed towards predicting no-pairs and one-pairs, then this could have resulted in additional Incorrectly clustered instances.

4 Neural Network Testing on Spambase

The neural network that was used for testing the data was WEKA's default MultiLayerPerceptron (500 epochs, 0.3 learning rate, 0.2 momentum, 10-fold cross-validation). The average of 3 iterations were taken for the data.

4.1 Running neural net on dimensionally reduced data

MLP Neural Network Metrics		
DR Algorithm	Model Building Time	Training Accuracy
None (w/o DR)	73.85 seconds	89.3636%
PCA	51.45 seconds	92.5886%
ICA	32.5 seconds	92.1104%
RP	19.75 seconds	90.4586%
Info. Gain	20.48 seconds	86.0248%

A note to make here is that model building time is different from training time, but as there was difficulty finding how to display training time instead of model building time, model building time was substituted. Training time can vary, though it seems to be multiplicatively longer than model building time in general.

On average, ICA seemingly performed the best on average in terms of performance and training accuracy, with PCA being the most accurate by a marginal amount but taking the longest time. PCA is expected in general to be more accurate in contrast to other dimensionality reduction algorithms, but when compared to RP which transforms data to a sparse matrix and information gain ranking features as opposed to calculating eigenvalues, eigenvectors, and generating an orthogonal matrix, much is to be desired in terms of time performance.

ICA's success for this can largely be attributed to the fact that the data for Spambase in general seems to overlap more with each other and are quite independent. The large number of attributes make it increasingly more important to be able to generalize based on individual and independent features.

Randomized Projection also did surprisingly well in little time, though it did not out-perform the original neural net by a considerable amount. The low time was expected due to the quick, random transformation to a sparse matrix as opposed to the long steps in other DR algorithms.

Information gain did not do quite as well in this case, which was surprising when considering how clustering and EM worked out well. A likely reason for this could be that the neural net weights did not seem to properly distribute accordingly with regards to the information gain dataset, and as a result generated outliers in the data that were unable to be generalized. In addition, a reduction of the hypothesis space could have attributed a moderate amount to the inability for Information gain to properly generalize.

4.2 Running neural net on clustering algorithm data

Two clusters were used to match the binary classifications of spam and non-spam. As a note as well, only one iteration was run for each DR + Clustering algorithm since each iteration took a rather long time to finish training.

MLP Neural Network Metrics w/ k-means		
DR Algorithm	Model Building Time	Training Accuracy
None (w/o DR)	73.85 seconds	89.3636%
PCA	88.67 seconds	99.0437%
ICA	38.88 seconds	99.4132%
RP	36.26 seconds	97.7396%
Info. Gain	38.92 seconds	98.1743%

MLP Neural Network Metrics w/ EM		
DR Algorithm	Model Building Time	Training Accuracy
None (w/o DR)	73.85 seconds	89.3636%
PCA	88.44 seconds	94.0882%
ICA	36.11 seconds	95.0011%
RP	25.88 seconds	94.8489%
Info. Gain	25.24 seconds	93.5014%

The model building time may be seen as deceptively low, but the training time takes upwards of half an hour for most of the DR algorithms, PCA in particular. One interesting observation here is the extreme improvement in accuracy as opposed to the prior counterparts and the original backpropagation neural network. There is a high likelihood that the data has been overfitted and cannot generalize special cases outside the 4600 instances of data observed.

In addition, ICA appeared to perform the best for the spambase data set, which seems to reverse the result obtained in the former dimensionality reduction observation, where it failed to cluster results appropriately. Somehow, this led to a great increase in accuracy when it came to adopting the neural network approach, which seems to suggest that the correct data was filtered from the noise. In spambase, the noise primarily lies in the amount of neighboring predictive attributes. EM in general is not as good as k-means for generating the centroid, as EM assigns rather than estimates, which makes k-means the big winner in this case.

5 Conclusion

Dimensionality algorithms can be used crucially to reduce the training time while maintaining and sometimes even improving the accuracy of training in the case of some feature selection algorithms. Depending on the dataset, k-means clustering is better at accounting for overlaps in numerical data through calculation of mean while Expectation Maximization seeks to group by higher probability through the use of maximization step.

Probability distribution within the data are also perhaps the largest determining factor in determining the accuracy of training data. Spambase is relatively kurtotic in nature, while the Poker Hand dataset is extremely skewed probability-wise, leading to long training times when attempting to predict accurately and inability to make accurate clusterings for the data.