CS4641: Unsupervised Learning

Paras Jain

1 Datasets

Two datasets were chosen Spambase and the 1984 Congressional Voting dataset, both from the UCI classification dataset repository. For convenience, the datasets are summarized below. For more details, refer to Assignment 1.

The voting dataset classifies 1984 congresspersons as Republican or Democrats based on their votes. The data was stored in ARFF format, which ABIGAIL can import. The data contains 16 discrete binary features and 1 discrete binary label class (democrat/republican).

The spambase dataset classifies emails from a training set of HP emails as Spam or Not-Spam based on various attributes, primarily relating to frequencies of key words, characters, etc. The data was stored in ARFF format, which ABIGAIL can import. The data contains 57 mixed features and 1 discrete binary label class (spam/not-spam). There are 4601 instances in this complete dataset.

2 Phase 1 Exploration of Clustering Algorithms

2.1 Background on clustering

Two main clustering algorithms were explored - k-means and expectation maximization.[1]

2.1.1 K-means

Given a set of data points, k-means clustering attempts to locate a fixed k clusters. A given point is assigned to the closest cluster centroid. This is implemented by choosing some random k points and let those be cluster centroids $c_1, c_2, \cdots, c_k$. Given k clusters, the new centroid is found and assigned as the centroid for the updated k clusters, now with centroid $c'_1, c'_2, \cdots, c'_k$. Repeat until convergence or iteration limit.

- **Consideration 1 - Choosing a good** k
  k is a fixed parameter and must be chosen carefully. This is the main lever for the machine learning practitioner using k-means. Too few clusters or too many clusters will sub-optimally encompass the data.

- **Consideration 2 - Choosing a good** $D(x_i, x_j)$
  For k-means, it is very important to choose a good distance measure. For this report, I am considering two measures Euclidean and Manhattan. Euclidean is the most often used distance measure traditionally.

2.1.2 Expectation Maximisation (EM)

The EM algorithm finds clusters by *a)* determining a function for the expectation for the log-likelihood and then *b)* maximizing parameters for the expected log-likelihood. This process again repeats to termination as with k-means. Higher values of log-likelihood indicate better choice of clusters.

[1] Clustering Algorithms are from WEKA, an open source machine learning tool, ABIGAIL was used for some Dimensionality Reduction algorithms

2.2 K-means distance function and optimal k-value selection based on error

We have two distance functions for k-means - Euclidean and Manhattan. This distance function is one of the key parameters for k-means.

Moreover, the choice of k is important. This choice will have significant impact on the performance of the clustering algorithm (perhaps even more than

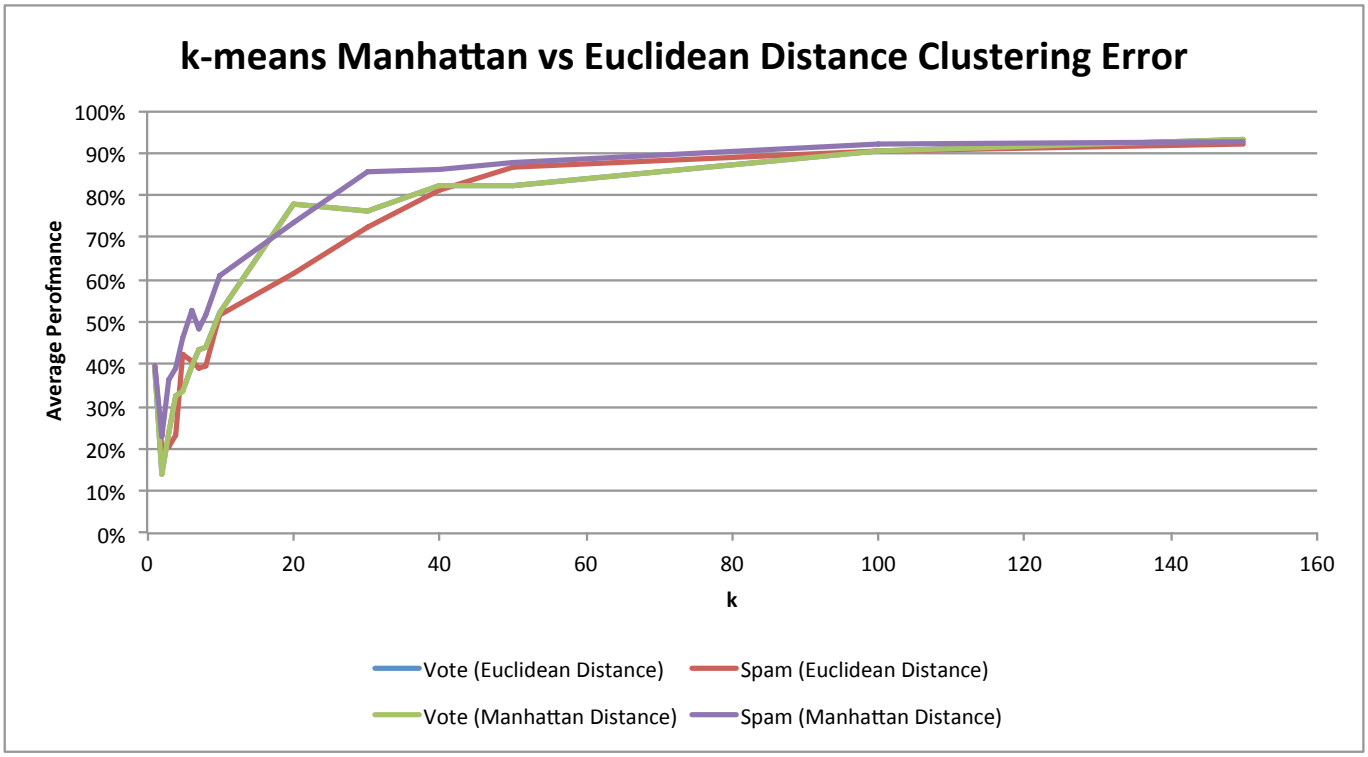

Figure 1: Error over various k values and distance measures

Analyzing the results of the experiment, displayed in Figure 1, we see that distance function had very little impact on the error for the voting dataset. However, for the Spambase dataset, the Euclidean distance measure had 2% better performance than Manhattan distance distance measure at best. Therefore, the conclusion is that the Euclidean distance measure is a better choice for these two datasets. This makes sense as it has a non-linear penalty for "farther" points.[2]

In Figure 1, it is immediately apparent that the optimal value for k is 2 as this is where error is minimized for both datasets. This result does make sense as each of the two datasets had binary classes. Interesting further study would be to repeat this analysis with a multi-class classification problem.

2.3 Elbow method of selecting k for k-means

A more information-theoretic approach to selecting the best k for k-means is the Elbow method[3]. The key fundamental idea behind the elbow method is to select a k value where the rate of change in the Sum of Squared Error (SSE) decreases abruptly. This is the point where the "percent of variation explained" (measured in part by SSE) gets less of a benefit from increased k. Obviously, SSE will decrease as k increases but the question is what k should we choose that is a tradeoff between smaller SSE and lower k.

In Figure 2, there are two clear elbows for the voting dataset - one at $k = 2$ and one at $k = 20$. Here, there is such a steep decrease from 2 to 3 that it makes sense to choose $k = 2$ for the number of clusters.

[2]Also one convenience of using the Euclidean distance measure is that the implementation of it in Weka is very fast.

[3]Implementation notes were taken from `https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set`

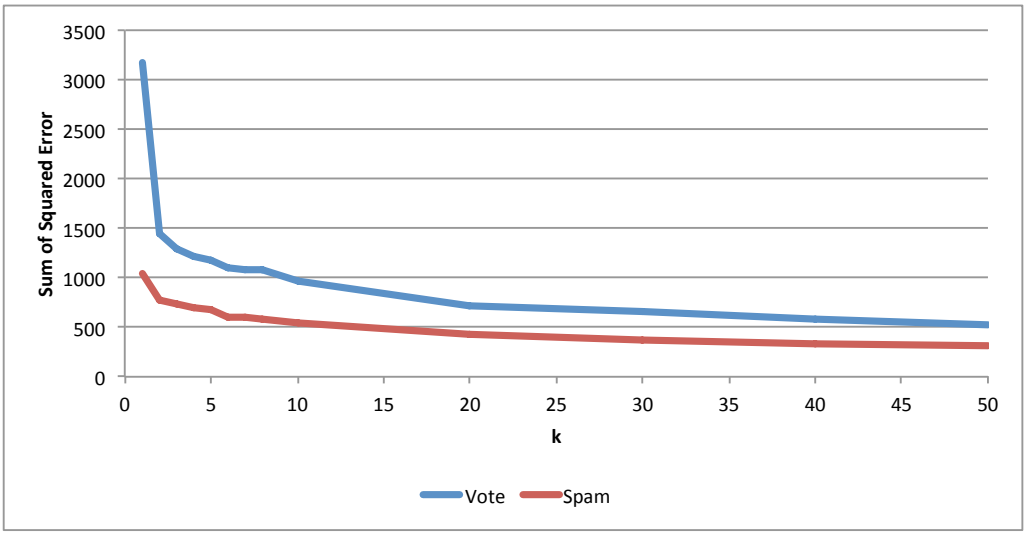

Figure 2: Elbow method for k-means for Spambase and Voting datasets

This makes sense due to the logic behind the elbow method from information theory. The choice is a little less clear for Spambase but a similar elbow (although less pronounced) is seen at $k = 2$.

2.4 Expectation Maximization - optimizing minimum σ

In short, it seems that for the test values of minimum standard deviation, no major difference was seen in the resulting performance for voting.[4]

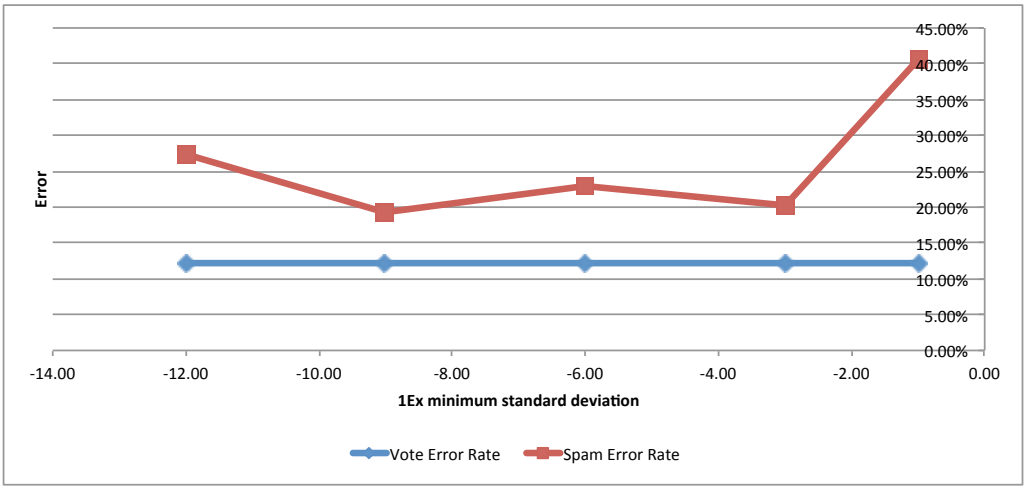

Figure 3: Log plot of error rate with varied minimum standard deviation for EM, with max clusters of 2

Referring to Figure 3, voting error rate was invariant to σ for EM. However, Spambase was sensitive to differences in σ for EM. This makes intuitive sense, however - the higher the minimum standard deviation is allowed to be, the less accurate the clustering needs to be and therefore this results in the higher error would be. The default values for Weka of around 1E-6 seem appropriate therefore.

[4]Raw results are included in the submission zip but for reader reference, voting was around 12% performance invariant to minimum σ

2.5 What do these clusters even mean?

When looking at these clustering algorithms, do these algorithms produce meaningful clusters? It seems that they do - the clusters for the voting dataset had only about a 12% error for representing the political party. Given that several attributes did not correlate well with class, this makes the result more significant. Playing around with the data in Weka visually, with 2 clusters for k-means, there is a clear democrat cluster and a clear republican cluster.

2.6 k-means vs EM - which is better?

To conclude phase 1 of analysis, I examine both k-means and EM to determine which one is "better". In this case, "better" is defined to be the algorithm with highest performance with optimized hyperparameters. For this analysis, optimal parameter values of $k = 2$ and $D(x_i, x_j) = Euclidean(x_i, x_j)$ were chosen for k-means. For EM, clusters were limited to at most $k = 2$ and the maximum σ was limited to 1E-9 which was the optimal value for both datasets.

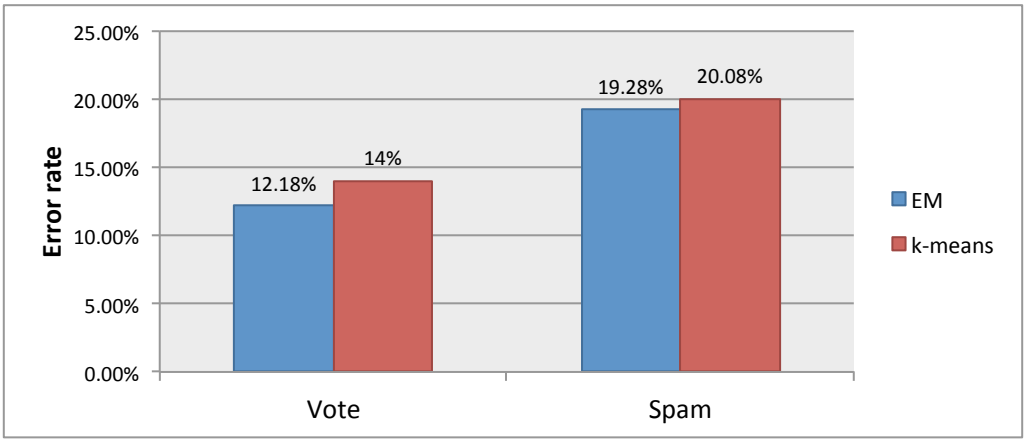

Figure 4: Comparison of minimum errors for optimal hyperparameter configurations for EM and k-means

As seen in 6, EM has lower error for both datasets. However, the delta between the two algorithms is so minor one can ask whether EM is indeed extracting some novel information in the process of clustering rather than converging more quickly. Indeed, by allowing k-means to continue more iterations, it begins to close the gap.

Depending on the distance measure, k-means has a bias to circular/spherical clusters. As EM explicitly accounts for uncertainty (soft vs. hard assignment), this may allow for a more flexible clustering which may account for the difference. Ultimately, the answer to which algorithm is that it depends on the situation and domain knowledge around the unsupervised learning problem.

3 Phase 2 - Exploration of Dimensionality Reduction Algorithms

3.1 Background on dimensionality reduction

3.1.1 Principal Component Analysis

PCA transforms attributes in order to create a new orthogonal coordinate basis which separates clusters farther apart. It uses eigenvalues as attributes with higher eigenvalues capture more information about the data. PCA is particularly interesting as it minimizes reconstruction error back to the higher dimensional space. This is due to PCA's process of maximizing variance captured along orthogonal coordinate axis.

3.1.2 Independent Component Analysis

ICA separates multivariate data into independent components that can be summed to reconstruct the original data. Each sub-component is a non-Gaussian[5] independent distribution. This means that there are no duplicate or co-variate attributes.

3.1.3 Randomized Projections

RP takes a n dimension space and projects it down into a k dimension space. RP preserves dimensions relatively well and is surprisingly effective given what it is doing. It is effectively trading a mathematically bound price of decreased accuracy for increased simplicity. This allows subsequent algorithms to run more quickly on the data given a known confidence error.

3.1.4 Random Subsets

Random subsets is a very simple dimensionality reduction algorithm that randomly selects a subset of attributes $s \subseteq S$ where $\|s\| = k$. I don't expect it to perform very well but it gives an simple baseline for comparison of dimensionality reduction algorithms.

3.2 Experiment 1 - PCA capture of variance and recovery accuracy

PCA's ability to capture variance with varying numbers of dimensions is interesting because of how PCA selects dimensions based on the largest eigenvalue.

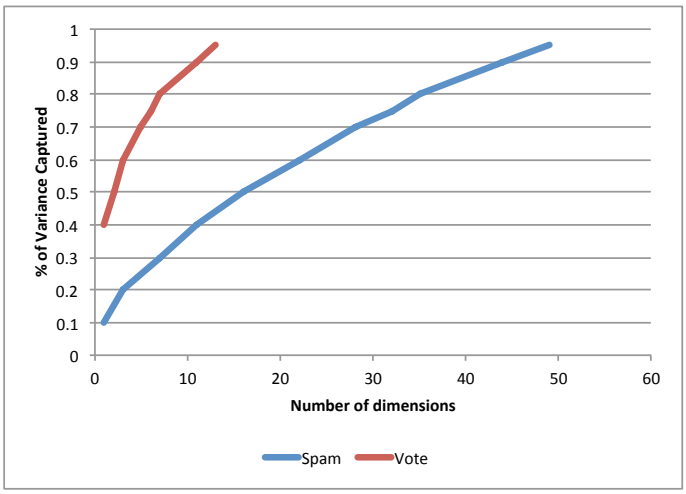

Figure 5: PCA's % capture of variance as a function of number of dimensions

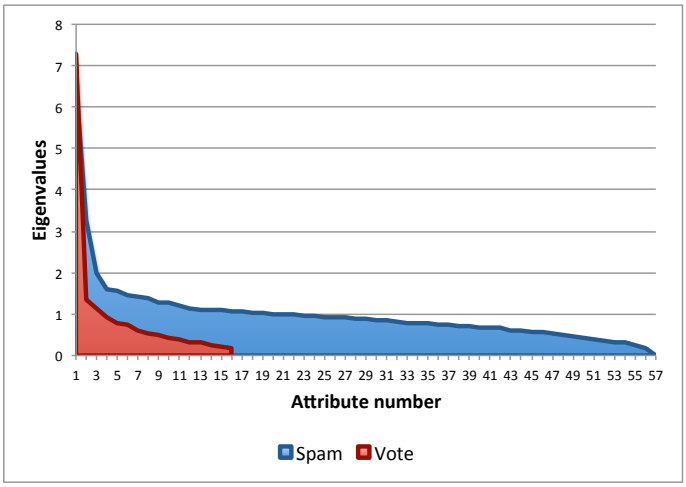

Figure 6: Distribution of Eigenvalues showing recovery of original distribution

Figure 5 is interesting as it shows that slightly less features than the number of original attributes are needed to capture almost all the variance. There is a non-linear relationship between variance captured and number of dimensions which does reflect the principle basis of PCA where there are certain axis of the transformed basis that are higher variance (say, the first transformed attribute) as compared to others (say, the last transformed attribute).

It is also interesting to note that the voting dataset converges more quickly than spam. Even when accounting for a fewer number of features, it achieves 40% of the variance with just one attribute as compared to the spambase dataset that requires 11 attributes. This is likely due to how predictably politicians voted along party lines which allowed for one or two attributes to capture more than half the variance of the data.

[5]Non-Gaussian ICA was used instead of MMI so feature selection depends on the kurtosis of attribute distribution.

What seems interesting as a result of Figure 5 is how PCA could be used to combat overfitting. It is effectively selecting the attributes that have the most variance, not dissimilar to how decision trees work using information gain. This overfitting consideration will be examined again later in the paper.

In Figure 6, it is interesting to compare the two distributions. For Spambase, there is a much longer tail. To reach 100% variance, the attributes map exactly to the original attributes. Therefore, this chart shows the share of varience captured by each attribute independently. It seems that the voting dataset had one attribute which encompassed a lion's share of the variance. The most prominent attribute for the voting dataset was primarily a combination of votes for el-salvador-aid, aid-to-nicaraguan-contras and other attributes that mostly seem to be foreign policy bills. This indicates that foreign policy as a topic was the most controversial type of bill. It is interesting how similar attributes cluster together into a single "topic" - this demonstrates dimensionality reduction as a technique to extract topics from say, a group of news articles.

3.3 Experiment 2 - ICA kurtosis analysis

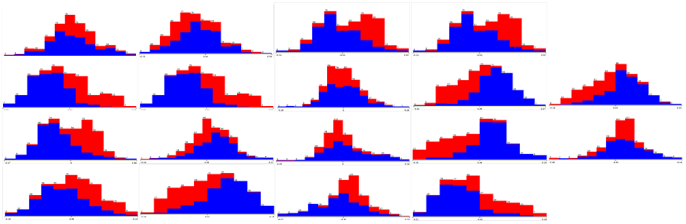

Figure 7: ICA feature distribution for Voting

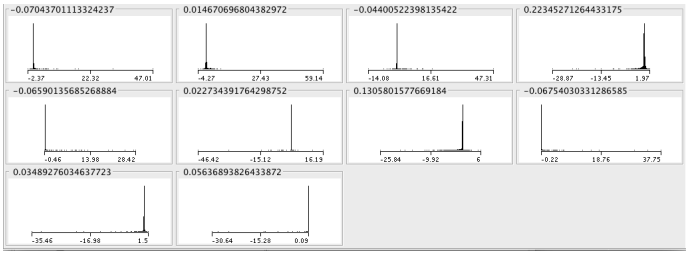

Figure 8: ICA feature distribution for Spambase

Voting					
Feature ID	1	2	3	4	5
Kurtosis	19.38	-1.74	4.70	4.35	3.00

Spambase					
Feature ID	1	2	3	4	5
Kurtosis	323	726	196	1246	1320

Table 1: Attribute kurtosis values after ICA dimensionality reduction

To analyze how Gaussian each attribute distribution is, the kurtosis[6] was calculated for both datasets - refer to Table 1 for the actual values for each of the reduced ICA attributes. For the voting dataset, several features were relatively close to being Gaussian distributions (see feature 5). However, spambase had many highly non-Gaussian attributes. This is apparent when looking at the visualization of the distribution in Figure 8.

Do the axis ICA produced make sense and do they represent something about the data? For voting, they do make sense as many of the original attributes are not mutually independent. This is likely why ICA produced one feature which was highly non-Gaussian while other attributes provided less independent signal about the data. However, spambase had many mutually independent attributes that each represented different classes of spam. Each attribute already was highly non-Gaussian with a large skew. This is because for word frequencies, the majority of instances had 0 frequency of the word. It seems that ICA produced meaningful axis therefore for both the spambase and voting datasets.

3.4 Experiment 3 - Randomized projection

As RP is random, there isn't a huge scope of interesting analysis to do here. I primarily looked into the question of how well RP was able to recreate the original dataset based on how small it reduced the dataset. In short, I wanted to examine the relationship between how much the dataset was compressed compared to

[6]Kurtosis is a measure of "tailedness" of a distribution where 3 is the kurtosis of a Gaussian

reconstruction accuracy. As RP is randomized, a total of 30 randomized trials were averaged for each test dimension.

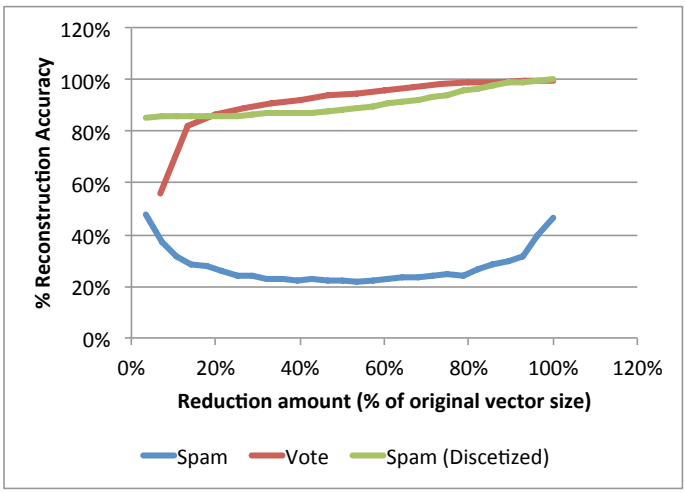

Figure 9: 30-trial average reconstruction accuracy for RP over various dataset compression amounts

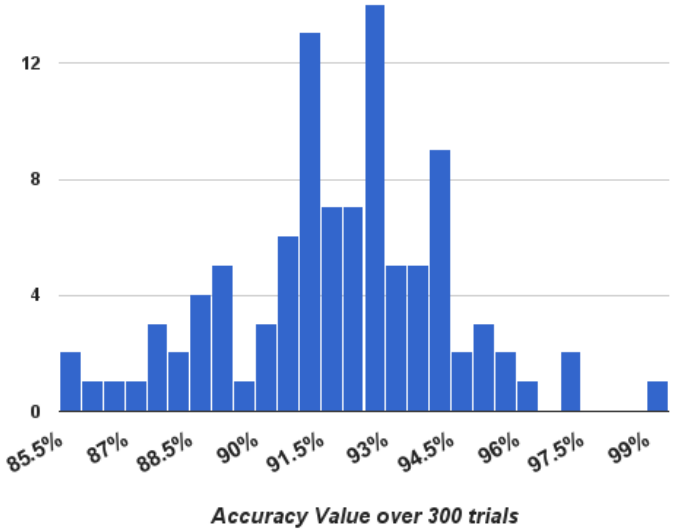

Figure 10: Histogram of variances for voting dataset over 300 trials

In Figure 9, vote and spambase perform very differently. For vote, RP is very effective at trading a small controlled amount of variance in exchange for a slightly smaller model. The relationship is almost nearly linear barring very small values of k.

For spambase, however, there appears is a very strange result where accuracy initially drops and then increases as k nears n. I suspect this is due to the difference between the datasets where voting is composed of binary attributes while spambase is composed of numerical attributes. I confirmed this was the case by discretizing the attributes of spambase and then running RP on the transformed dataset. In Figure 9, the discretized spambase dataset is almost nearly linear and has generally very high performance.

This experiment shows how RP is designed to trade off a small amount variance for a smaller number of attributes. It is effective at reducing the size of instance spaces with discrete attributes.

To study how random RP is actually, I ran 300 trials of RP over the voting dataset and examined the distributions of reconstruction accuracy. Refer to Figure 10 for the full distribution but the key insight is that it is a pretty right bell curve with the majority of values in the 91-94% range. It appears that variance is bounded reasonable for the voting dataset - this demonstrates that while RP is actually random, individual results do not vary too greatly.

3.5 Experiment 4 - Random Subsets

There isn't too much interesting analysis to do on random subsets given how simple it's algorithm is. RS provides a good baseline for the experiments using NN later. Reconstruction error is difficult to study as attributes are just dropped in the process of reduction to a smaller dimension.

4 Phase 3 - Exploration of Clustering after Dimensionality Reduction

For Figure 11 and Figure 12, note that RP and RS are best error rates over 3 random trials.

I was particularly interested in evaluating the effect of dimensionality reduction on the results of clustering. Refer to Figure 11 and Figure 12 for the misclassification errors for k-means and EM over No Dimensionality Reduction, PCA, ICA, RP and RS.

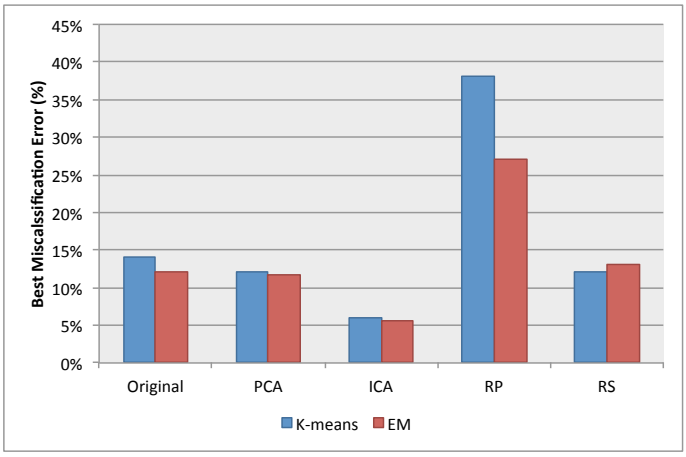

Figure 11: Error rate for k-means and EM after dimensionality reduction for voting dataset

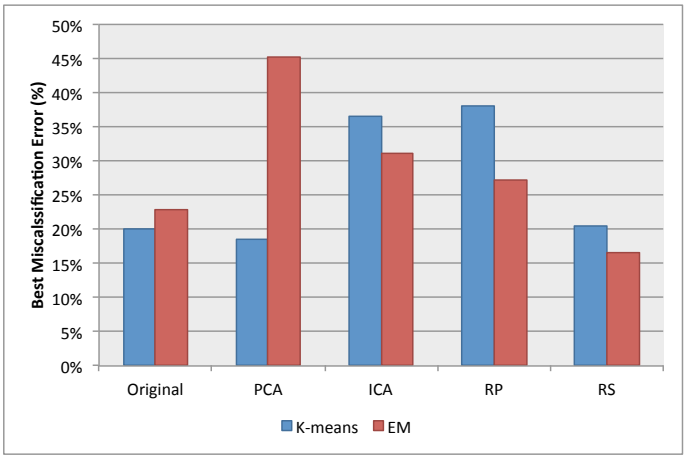

Figure 12: Error rate for k-means and EM after dimensionality reduction for spambase dataset

These results are very interesting as the impact of clustering was very different between the two datasets. For voting, it seems that PCA, ICA and RP all improved clustering performance. I suspect that this is due to the dimensionality reduction removing noise in the form of redundant attributes which were common in the voting dataset where each attribute provided redundant signal about political party. RP did not seem to perform very well compared to RS which is surprising - I suspect that is because RP functions by trading some given amount of variance for a smaller attribute which almost ensures that it will not perform significantly better than the untransformed data.

For spambase, the dimensionality reduction algorithms had performance across the board. Surprisingly, RS managed to improve the error rate when I expected it to perform the worst as a baseline. For the other algorithms, performance dropped significantly. I suspect this is because the spambase dataset did not have significant duplication of attributes. By reducing the attribute size for spambase, valuable signal was being discarded. I suspect these algorithms could not find some combination of attributes to preserve variance as each attribute of spambase provided signal about some diverse type of spam.[7]

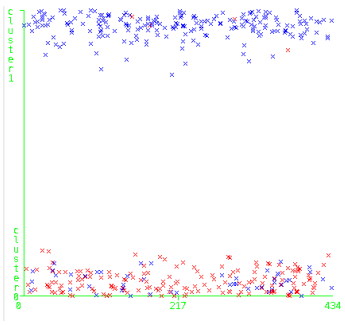

Figure 13: K-means cluster assignments before PCA for Voting

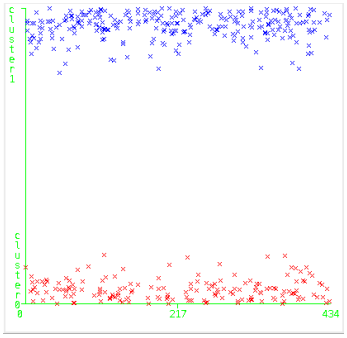

Figure 14: K-means cluster assignments after PCA for Voting

After running PCA[8] and then k-means, it appears that dimensionality reduction did not have a significant impact on cluster assignments - refer to Figure 13 and Figure 14 fora visualization of cluster assignments. Only minor changes occurred where misclassifications actually reduced in a few select cases. Similar results

[7] e.g. "money" vs "viagra" vs "HP" vs phishing which all have separate attributes that provide signal
[8] PCA with 70% variance capture

were seen for the other algorithms. This indicates that dimensionality reduction can potentially help extract signal from noise which can help counter the curse-of-dimensionality. PCA actually performed very well when paired with k-means on the spambase dataset. This is not surprising as the new orthogonal basis PCA produced would make the clusters produced by k-means "cleaner".

4.1 k-means vs EM after clustering

For the voting dataset, EM and k-means had very similar performance for the most part. However, spambase had significant differences between k-means and EM for PCA. This is explained above but I suspect that spambase's many mutually independent attributes is complemented well by PCA for producing clean clusters with maximal distance between centroids which is why k-means performs so well. Ultimately, the performance of k-means and EM seem to be similar but the choice of which one is better depends on the dimensionality reduction technique and the characteristics of the dataset. For most cases, the two clustering algorithms seem to perform similarly.

5 Phase 4 - Training a NN on Reduced Data

After training a neural network using the reduced spambase data, performance and runtime information was captured.[9] See 15 for the results of this phase of the project. Spambase was chosen for this analysis as voting already performed very well using NN versus spambase. The unreduced dataset performance is included as well as a baseline.

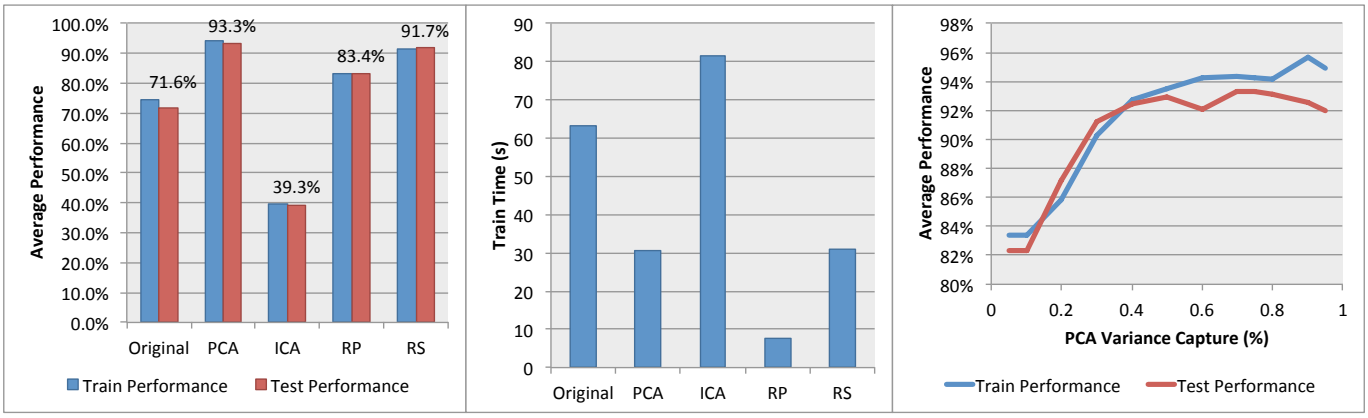

Figure 15: *Left:* Train/Test performance for NN trained on reduced spambase data, *Middle:* NN training runtime for various reduced spambase datasets, *Right:* Train/Test performance of a NN trained on reduced spambase data as a function of PCA's variance capture

It appears that the reduction algorithms produced varying results. However, it is surprising as PCA, RP and RS each produced higher performing models as compared to the unreduced model. PCA, RP and RS also each took significantly less time to train as compared to the baseline. Unexpectedly, ICA took quite a bit longer train than the unreduced dataset.

I suspect ICA performed poorly here as the number of attributes increased when extracting all the sources. This led to a higher runtime with worse performance as NN will require more iterations to converge. Retraining the NN with 2000 iterations, we get an accuracy of 45.72% which is a bit closer to the rest of the algorithms yet still worse than the baseline. It likely will take many iterations for ICA to converge towards the other datasets, if even ever.

I wanted to see what impact the degree of dimensionality reduction impacted performance which is shown on the right of Figure 15. This chart is very interesting as it shows the effects of overfitting. This may explain why dimensionality reduction actually can improve test performance as compared to an unfiltered

[9] 500 training iterations

dataset. Test performance continues to rise as more PCA variance is captured - however, test performance is maximized at 70% of the variance being captured. As more variance is retained, test performance actually decreases - this is a evidence of some form of overfitting occurring.

6 Phase 5 - Training a NN on Reduced and Clustered Data

EM and K-means were applied to the results of the spambase dataset after dimensionality reduction. Each clustered, reduced dataset was passed into a MultilayerPerceptron in Weka. The results are displayed in Figure 16.

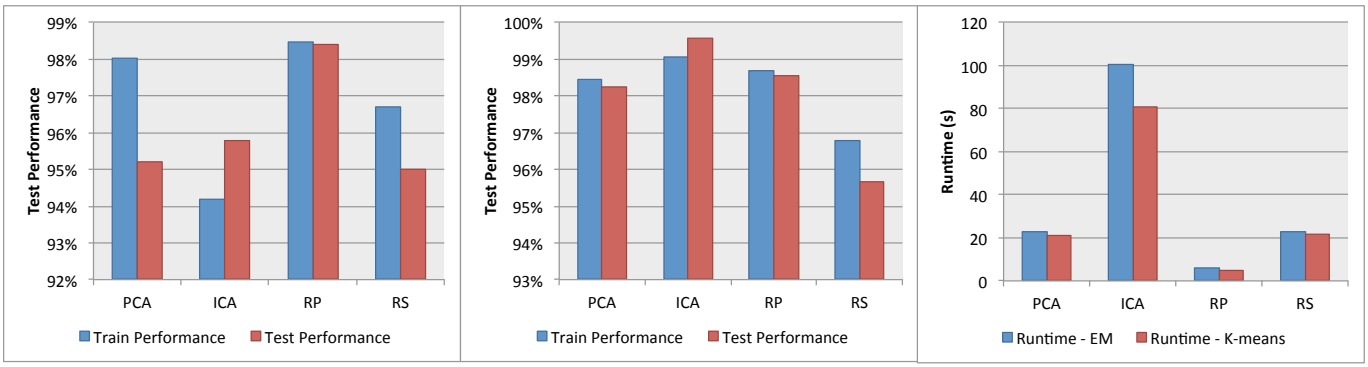

Figure 16: *Left:* Train/Test performance for NN trained on EM clustered, reduced spambase data, *Middle:* NN training runtime for K-means clustered, reduced spambase datasets, *Right:* NN training runtime

It is important to note the scales on the performance charts which are all above 90% - all of these algorithms performed very well regardless of clustering algorithm or dimensionality reduction algorithm. Any differences we are observing here are over a few percentage point differences. At these microscopic scales, some of the effects we observe could be attributed to factors such as overfitting or how we are partitioning the train/test datasets.

This is ICA's time to shine! When paired with k-means, clustered ICA had the highest test performance at near 100%. This is far better than the original dataset performance of 71%. This did come at the cost of a significant runtime penalty when compared to other dimensionality reduction algorithms. It seems RP also had better performance after clustering rather than passing it's results directly into a NN.

K-means seemed to have significantly better test performance than EM on the reduced datasets across the board. It appears that EM produced variable results that did not necessarily generalize to test datasets (as seen with PCA or ICA). Compare that with k-means which excels. Referring to Phase 1, however, we see that k-means performed better on the original datasets. This may be the reason that k-means again performs well on the transformed datasets.

I find Randomized Projection with K-means to be the winner ultimately for spambase classification. RP is able to accomplish a very high test performance when paired with K-means with an extremely fast training runtime. One could run many trials with RP and select the highest performing model and this randomized trial approach would still likely have better runtime than other approaches.

7 Conclusion

Dimensionality reduction and clustering algorithms are great ways to improve the runtime of machine learning problems. Some DR algorithms are faster and slower so it depends on the structure of the data and the ML algorithm being used. In some cases, accuracy improved which as shown by PCA parameter tuning, may be a way to reduce overfitting. Choice of Clustering/DR algorithms depends on the format of the dataset (discrete vs continuous, num. of attrs., missing data, independence of attrs., etc.). However, as shown in Phase 5, combining DR and Clustering can produces more accurate models while reducing runtime.