Brief Description/Reminder of What the Data Sets Are:

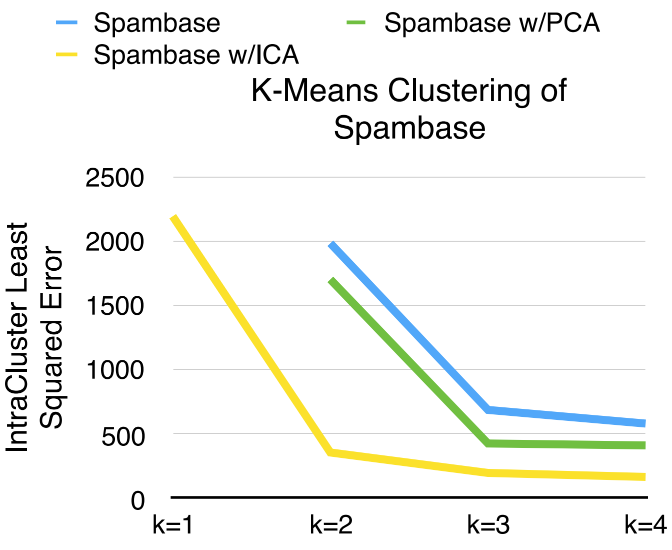
I used the Spambase dataset that I used for project 2. Spambase is a binary classification task; the classification decides whether or not an email is spam. There are 57 numeric attributes that describe the words in emails, and there are only 4600 examples, so this dataset is very sparse. The distribution of the values of the attributes is heavily skewed, as 0 is the most common value for most attributes, and it probably has a high kurtosis. Spambase is interesting because through simple analysis of the word frequencies in the emails, algorithms can accurately predict (>90%) whether or not an email is spam.

I used the Krkopt dataset that I used for project 1. Krkopt contains chess endgame positions for games containing a white king and rook against a black king where it's black's turn to move (Krkopt). There are six features: the rank and file of the white king, the rank and file of the white rook, and the rank and file of the black king. For the purpose of classification, we assume that both sides will play optimally, and we wish to use the features to predict the number of moves it will take for white to win. Krkopt has 18 different potential classes, and the class attribute is nominal. ICA was unable to run on this dataset because Krkopt has multiple nominal attributes, positions on the chessboard.

How did I choose which k to use?

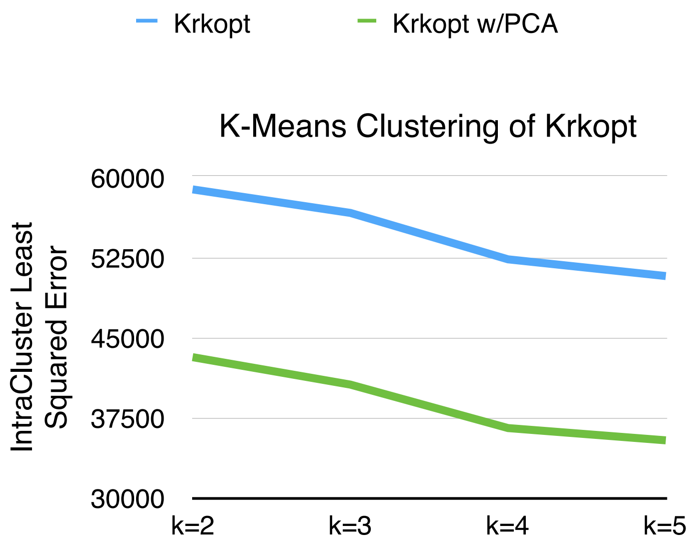
I used the Elbow Method to decide which k to use for k-means clustering. Unsupervised learning is in many cases ambiguous, as the problem of data description is much less well-defined than the problem of function approximation. It thus requires human interpretation, and the elbow method is an effective heuristic for determining the appropriate number of clusters in a dataset. As one increases the number of clusters, the data is continually described better. For k-means clustering, the average least squared error around the centroids of cluster will continually decrease as k increases and the data is better described. The elbow method allows one to determine the appropriate k by looking at the derivative of the decrease in least squared error. The point at which the derivative of least squared error sharply decreases in magnitude, and adding another cluster does not contribute significantly to describing the data better, is the point at which one should stop increasing k.

As one can see in the chart, for the Spambase data set, Spambase after PCA had been applied, and



Spambase after ICA had been applied, there are sharp decreases in the rate of change of the intracluster least squared error at the point of k=3 for Spambase and Spambase with PCA applied, and at the point k=2 for Spambase with ICA applied.

Similarly, for Krkopt, I noticed that there was a decrease in the the magnitude of the derivative by 2x to 3x at the point k=4 for the normal Krkopt data set and for Krkopt with PCA applied. The elbow method is subjective in some ways, and requires interpretation of the data, but it is a useful heuristic nonetheless.



How did I choose to use Euclidean distance rather than Manhattan?

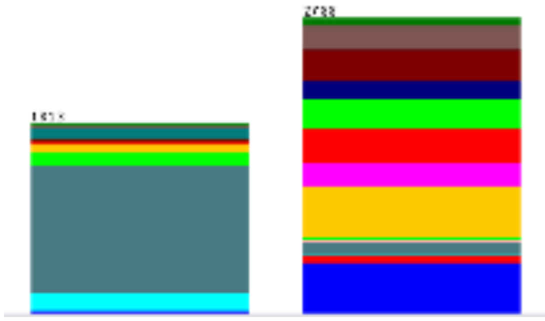
I decided to use Euclidean distance rather than Manhattan distance for Krkopt. This may seem like a controversial decision, as a chessboard is made up of adjacent squares laid out in a grid, and the rook must move in straight lines in either the horizontal or vertical directions, which is akin to Manhattan Distance, but may not move diagonally. However, I decided that because the rook can move as many squares as it wishes in the same direction in a turn, I decided that Manhattan Distance was not a good way to measure the rook's proximity to a location. Furthermore, the king can move in any direction, even diagonally by a single square at a time. I decided that the location and movement of the king was approximated better, though still imperfectly, by Euclidean distance, as it can make a diagonal move in a single turn, and not two, as would have been calculated by Manhattan Distance.

As for Spambase, the decision was mostly arbitrary, as all of the attributes are real-valued and numeric. Euclidean distance is more sensitive to large discrepancies between points for a single attribute even if all of the other attributes are very similar, whereas Manhattan Distance, absolute distance, is a better reflection of overall differences between data points. By choosing Euclidean distance, I decided that it was more likely that a few attributes would have especially large usefulness in predicting spam than that spam emails would just be generally different for most attributes.

Description of Spambase Clusters:

For Spambase, using the elbow method, I found that the best k for k-Means Clustering was k=3. Unfortunately, k-Means ended up splitting all of the spam into a single cluster, such that neural network classification when ran on those clusters gave an accuracy of 100%, as the neural network quickly discovered that there was an attribute that perfectly reflected the correct classification of the example.

Expectation maximization assumes that there are hidden variables in the data set, and iteratively guesses the distribution of those variables. Expectation maximization split the dataset into 16 clusters and gave results that were not obviously invalid, but were still questionable, given how high the percentage classification was. The neural network performed with an accuracy of 99.4928% at 500 iterations. In the graph, the colors represent clusters and the two bars are the two classifications in the problem. As can be seen, the clusters do not seem to separate on the basis of class.



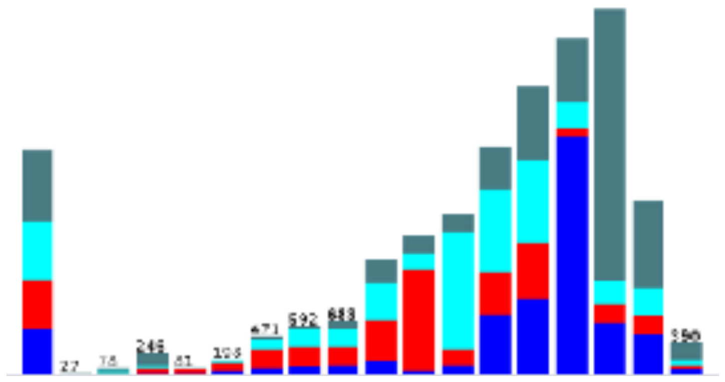
PCA mixed with k-Means with a k value of 3 gave an accuracy of 100% at 1500 iterations, as the clusters once again split the classifications perfectly, and immediately caused overfitting in the neural network. A neural network classifying Spambase transformed by PCA gave 92.8986% accuracy at 2500 iterations. This seems to be a continual problem for K-Means, I think I should change the algorithm to leave out the class so as to describe the underlying structure of the data without regard to class.

Spambase transformed by ICA clustered with K-Means with k=2 also trivialized the problem, as the clusters were equivalent to the classification values and the neural network achieved 100% performance.

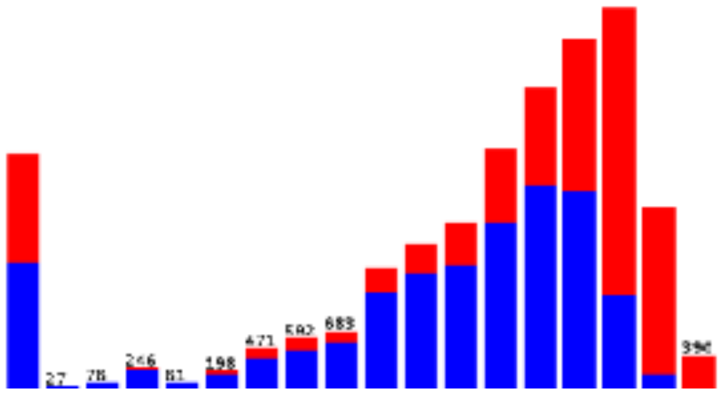
Spambase transformed by ICA and clustered with Expectation Maximization had the dizzying classification accuracy of 99.4203% at 1500 iterations of a neural network, which I'm also skeptical of, but appears to actually represent some aspect of the data. Expectation maximization seemed to consistently work very well and express interesting aspects of the data

Description of Krkopt Clusters:

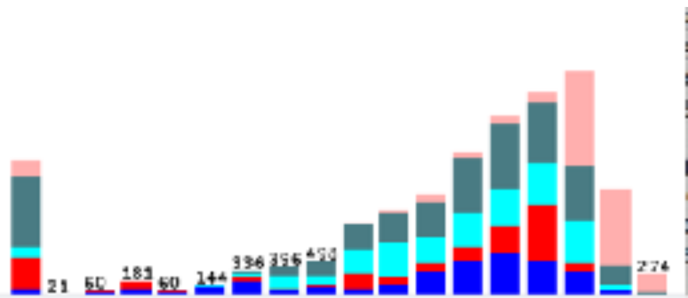
For Krkopt, using the elbow method, I found that the optimal clustering for K-Means Clustering is at the point k=4. K-Means clustering works by picking random central points and partitioning the data by how close the points are to that center, then iteratively moving the center to the average of the points in that cluster, then re-calculating which points should be in which cluster based on the new center. It's difficult to categorize the shapes of the clusters in respect to different attributes, as it varied significantly, with some being completely flat and others having normal distributions. However, K-Means was an incredibly effective way to increase neural network performance, and does not appear to have explicitly fallen prey to the problem of trivializing the problem by simply replicating the classifications, as there are 18 possible different classifications and only 4 clusters. However, the accuracy of the neural network trained on 70% of the data with 30% used for testing with 1500 iterations was 99.3109%, which seems to be inconceivably high. The clusters in the image are represented by the colors, whereas the classifications, starting with draw, then ranging from one move to sixteen moves to checkmate are along the x axis.



Expectation maximization on Krkopt performed similarly well, splitting the dataset into only two categories but drastically improving neural network performance to 99.0733% at 500 iterations. Similarly to the previous image, the clusters are the colors whereas the classifications are along the x axis. Once again, the clusters do not on first glance seem to trivialize the problem, but do lead to ridiculous, hard to believe outcomes in terms of classification accuracy. Expectation maximization seems to tend to create fewer clusters than K-Means, but automatically find a number of clusters that well describes the data set and helps to improve classification accuracy.



For Krkopt after the dataset had been transformed by PCA, I ran expectation maximum

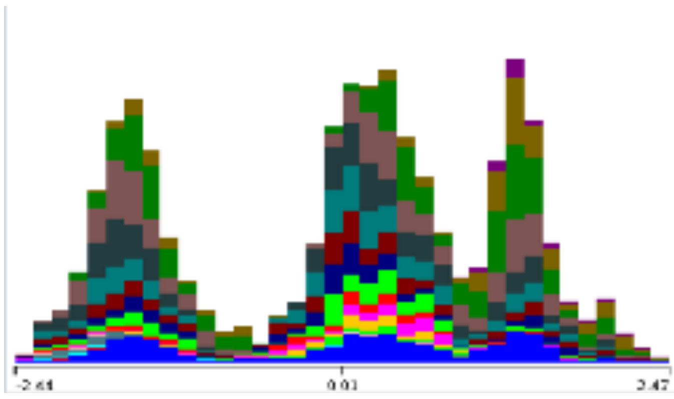


clustering with a maximum of 5 clusters. Expectation maximization assumes that there are hidden variables in the data set, and iteratively guesses the distribution of those variables. The classification accuracy of a neural network with 500 iterations was 97.9446%.

For Krkopt after the dataset had been transformed by PCA, I used the elbow method to find the appropriate number of clusters then ran K-Means Clustering on it with k=4. I received very similar performance when classifying using a neural network. With 1500 iterations I got an accuracy of 98.2298%. This is not even comparable to the performance of a neural network on Krkopt after PCA had been ran on it, which with 1500 iterations had an accuracy of only 52.6435%.
Neural networks were ran against every combination of clusters and feature transformation algorithm described, each time with 70% of the data used for training and 30% used for testing, with 500, 1000, and 1500 iterations.

Principal Component Analysis of Krkopt:

PCA did a very poor job of reconstructing the Krkopt data as it had many nominal attributes. The Krkopt dataset uses only 6 attributes, the locations of three pieces on the board, to model a very complex problem. When I ran PCA without a constraint on the maximum number of attributes, various linear combinations of the 6 attributes were projected into 19 dimensions. The attributes describe some of the key combinations of nominal attributes that PCA was unable to transform numerically, such as when the white king is close to the black king, and when the black king is close to the white rook.
As PCA seeks to maximize the variance across each dimension, and gives an ordered list of properties, the first two attributes have the highest standard deviation, and are linear combinations of various positions of the white king. The next data attributes relate the positions of the white king and black king, then most of the following attributes relate the rook to the position of the black king in various ways. Only a few of the attributes appear to vary according to normal distributions. Many appear to have multiple bell-shaped curves and a very low kurtosis such as in the graph pictured.

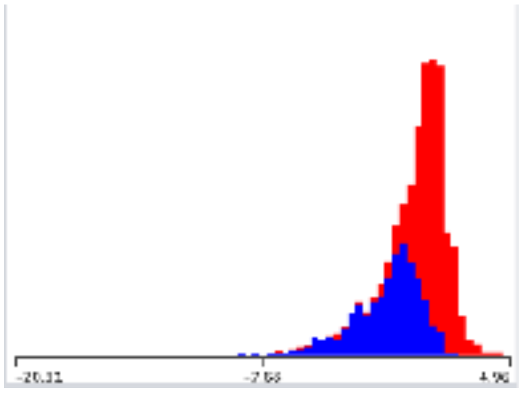


PCA decreased the performance of neural networks on Krkopt. A neural network ran for 1500 iterations on the original Krkopt dataset achieved 58.0% accuracy, whereas a neural network ran for the same number of iterations on the PCA version had only a 52.64% accuracy. I think this is due to the high number of values possible for nominal attributes and PCA's problems dealing with them. I think this could be fixed by transforming the alphabetical attributes found in Krkopt to numeric attributes.

Principal Component Analysis of Spambase:

PCA was very useful on the Spambase dataset. All of the new attributes created by PCA appeared to be very similar to normal distributions, and had high kurtosis. It reduced the number of attributes from 57 to 48. As the attributes are ordered by maximum variance, the first and most important attribute was the word frequency of a few words. The next most important attributes seemed to have the highest weights when used in the neural network, and were related to the longest running sequence of capital letters.

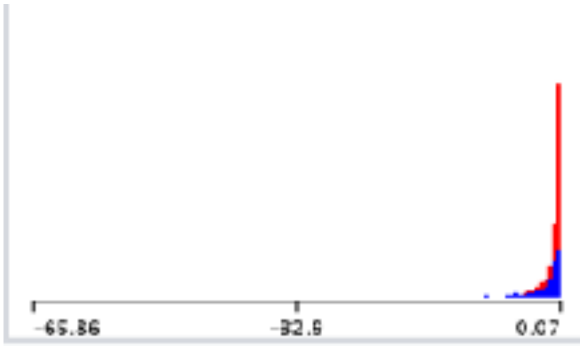
The graph pictured is of the dataset's variance along the axis of longest running sequence of capital letters and total occurrence of capital letters. The algorithm identified many attributes in which the spam was often on one end of the bell curve, and the non-spam was on the other end. Note that the spam is red, and non-spam is blue in the graph.



A neural network trained on the original dataset had 92.53% accuracy and only converged after 3000 iterations, whereas a neural network trained on the dataset with PCA applied became effective much faster with 92.6% accuracy at 500 iterations and achieved a maximum of 92.9% accuracy at 2500 iterations.

Independent Component Analysis of Spambase:

Independent component analysis seeks to split a set of attributes in a dataset into a new set of statistically independent attributes and discover the underlying hidden variables behind the various components. ICA assumes that there are independent sources of information that mix to cause the attributes in the dataset. Interestingly enough, for Spambase,



when ICA was not constrained to pick a certain number of sources of information, it projected the 57 attributes of Spambase into a space of 57 sources. Many of the sources appear to be normally distributed, but still more of them appear to be a single half of a normal distribution, with a high peak and outliers on only a single side of the distribution, as pictured.

ICA significantly harmed the performance of neural networks on Spambase, causing an accuracy of only 67.9% at 1500 iterations of the neural network. I think this is because there is likely to be a much lower number of independent signals to parse out of the data than 57, so ICA ended up just distorting the attributes in various ways to fit normal distributions. Overall, for this dataset, ICA appeared to be less useful than PCA at reconstructing the data. It found 57 statistically independent signals from 57 attributes, but seems to have mangled the data in doing so. Thus, I think the 57 attributes are already statistically independent, which would explain why ICA was not useful for this dataset.

Analysis of Randomized Projection on Spambase:

Random projection on Spambase was surprisingly effective at reconstructing the data with fewer attributes. Spambase has 57 attributes, but when these 57 attributes were randomly projected into 10 dimensions, a large amount of data was retained. This is evidenced by the classification accuracy of a neural network running 500 iterations on the newly transformed data. Across five different random projections, the average classification accuracy was 79.98% with a standard deviation of 5.21. I'm a little bit shocked at this, but it makes sense that the reconstruction of Spambase was much easier than the reconstruction of Krkopt, which includes many nominal attributes.

Analysis of Randomized Projection on Krkopt:

I tried projecting the attributes of Krkopt into 4 dimensions and 5 dimensions using RP, and tried five trials of each. For RP with 4 dimensions, the data was very poorly reconstructed. Many attributes had very low kurtosis, and the shape of the data was quite flat. When I used a neural network with 1500 iterations to predict the optimal win depth of the data after RP, the results varied between 23.85% accuracy and 28.2% accuracy, with a standard deviation of 1.75% and an average performance of 25.3%. This was a horrifically low reconstruction rate, as a neural network achieved 58% accuracy before the projections were performed. RP into 5 dimensions performed similarly, if a little worse, with an average of 24.4% and a standard deviation of 2.27%. RP with 5 dimensions consistently seemed to come up with bell-curves rather than the flat distributions, but this did not improve performance.

Analysis of Linear Discriminant Analysis on Spambase:

LDA finds projections that makes it easier to discriminate based on the labels of the data. LDA makes the simplifying assumption that each attribute has a gaussian distribution and then determines the mean and variance among each attribute for each class. For Spambase, there are only two classifications, spam or not spam. LDA found the mean and variance for each attribute for those two possible classifications, then uses that data to probabilistically estimate how to classify new examples. After being trained against 70% of the data, when tested against a testing set of 30% of the data, LDA had a surprisingly high 88.269% accuracy at classifying spam vs not spam.

I’ve also learned that nominal attributes are the bane of all dimensionality reduction algorithms, as once again Krkopt does not work with LDA.