Yukt Mitash
Professor Hrolenok
CS 4641: Machine Learning
April 7, 2019

Assignment 3: Clustering and Unsupervised Learning

Introduction: In this assignment, we will be exploring two clustering and three dimensionality reduction algorithms. The clustering algorithms are k-means clustering and Expectation Maximization. The dimensionality reduction algorithms are Principal Component Analysis, Independent Component Analysis, and Randomized Projections. These algorithms fall into the field of Unsupervised Learning. This is different from a lot of our past work in the sense that there is no training or testing data set. There is just one large data set. The purpose of these algorithms, primarily the clustering algorithms, is to designate a specific group for each data point without ever seeing the target variable of any of the data sets. The purpose of the dimensionality reduction algorithms is to use linear algebraic methods to reduce the number of features of the data sets while losing the smallest amount of information. First, we will run the clustering algorithms on the two data sets mentioned below. Then we will run the dimensionality reduction algorithms and see how the clustering algorithms behave with the data. Finally, we will run Neural Networks on the data after having done dimensionality reduction. All algorithms were implemented using the SciKit learn Python library.

Data Sets Review

Data Set 1 Spotify Song Attributes: This data was collected by someone using Spotify's API. Essentially, it contains 2017 songs with the following information on each song: Acousticness (confidence measure of 0.0 to 1.0), Danceability (confidence measure of 0.0 to 1.0), duration (in milliseconds), energy (confidence measure of 0.0 to 1.0), instrumentalness (confidence measure of 0.0 to 1.0), key (an estimated overall key with integers mapping to keys using standard pitch class notation), liveness (confidence measure of 0.0 to 1.0 of whether or not the track is live), loudness (measured in decibels averaged across the track form -60 to 0), speechiness (the amount of speech used in the track measure form 0.0 to 1.0), valence (measure from 0/0 to 1.0 of positivity), tempo (estimated in Beats per Minute), modality (major is represented by 1 and minor by 0), and time_signature (a measure of meter). In addition, each entry in the data contains a classification value of 1 or 0, respectively pertaining to whether the user liked the song or not. All algorithms for this data set will focus on learning/predicting whether or not the user liked the song based on the 13 above characteristics. All data pertaining to a given song is determined by the Spotify API which one can learn about here: https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/. The data set is found on kaggle.com and you can learn about that here: https://www.kaggle.com/geomack/spotifyclassification. This data set is very interesting because it examines the makings of how a single person can come to like or dislike a song unconsciously because of 10 different features. The purpose of using this many features is to see how the algorithms perform classification when given many attributes to view the data by.

Data Set 2 Graduate College Admissions: This data was posted on kaggle and includes the following features: GRE Score, TOEFL Score, University Rating (rated from 1-5), SOP (rated from 1-5), LOR (rated from 1-5), CGPA, Research (1 or 0 respectively pertaining the whether or not an applicant had done research), and Chance of Admit (represented as a decimal). The size of the dataset is 500. To turn this into classification problem from a regression problem, the algorithms must be testing/predicting a binary output (admit or not). To do this, I modified the data set so that all chances of admission greater than or equal to 0.725 were represented by a 1 and all chances of admission below that were represented by a 0. Rather than predicting whether or not an applicant with certain scores will be admitted or not (a difficult task), algorithms will be predicting whether or not an applicant has a chance of being admitted that is greater than or equal to 0.75. In addition to this, I wanted to change the structure of this problem from the last one to examine how the algorithms perform under different conditions. To do this, I decided to only look at three factors: CGPA, GRE score, and University Rating. This allows more variety across problem sets. Learn more about the data here https://www.kaggle.com/mohansacharya/graduate-admissions. This problem is interesting because it examines how one can use machine learning to turn college admissions into a concrete mathematical question rather than the subjective one it is currently.

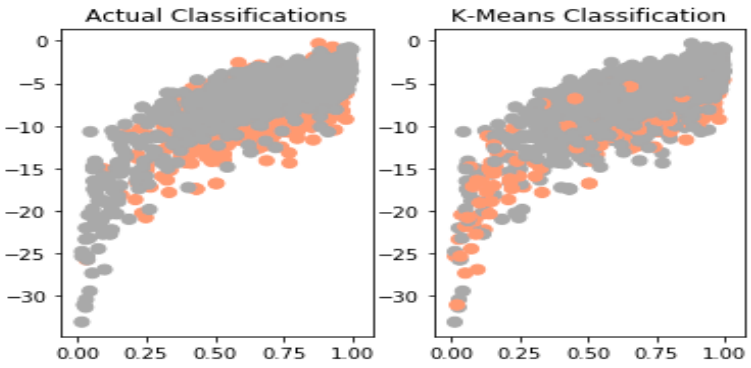
Clustering

Clustering is one of the main techniques of unsupervised learning. Essentially, when given a large data set, it uses mathematical and statistical techniques to categorize all points according to their features into a certain grouping. This technique is interesting for classification problems because the algorithms do not understand which features correspond to which classification, they are simply able to separate points from each other based on features. The purpose of these algorithms are not to assign data to specific outputs, but rather to make sense of unlabeled data. The two clustering algorithms we will examine are k-means clustering and Expectation Maximization. Unfortunately, the scikit learn library does have a function for determining the accuracy of clustering models based on their centers for K-Means clustering. I decided to write a function that calculates the Euclidean Distance in n-dimensions of each point to its cluster's center. In layman's terms, this is the sum of the squared error for each point and its center. Euclidean distance is an appropriate measure for the accuracy of k-means because k-means operates at each iteration using Euclidean distances to reevaluate each grouping. I will be using the average of each point's Euclidean Distance to its center to judge the accuracy of each model for k-means clustering. For Expectation Maximization, each cluster is represented by a Gaussian distribution. Therefore, it would make more sense to evaluate the models' accuracies with statistical techniques.

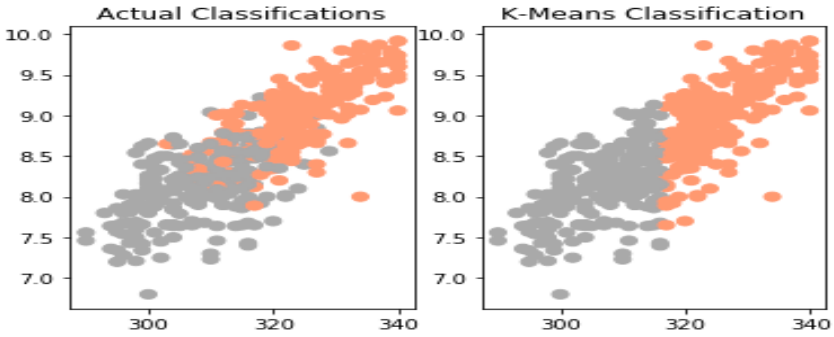
K-Means Clustering

Introduction: K-Means clustering is a simple, but effective clustering algorithm. The algorithm is given a set of points in n-dimensional space and a value for n (this will be the number of clusters). First the algorithm randomly picks k "centers." These are points in the plane that contains the data set. They do not necessarily have to be actual data points. Then, each point is put into a group based on the center they are closest to. After each point is in a group, the centers of each group are recalculated based on an averaging of all the points in the group. After this happens, each point is once again categorized as belonging to one of the k groups. The centers are recalculated and this process continues until the centers converge, or stop changing value between iterations. This algorithm is like Randomized Hill Climbing, in that the algorithm is always seeking to improve based on its neighbors.

Data Set 1 (Songs):



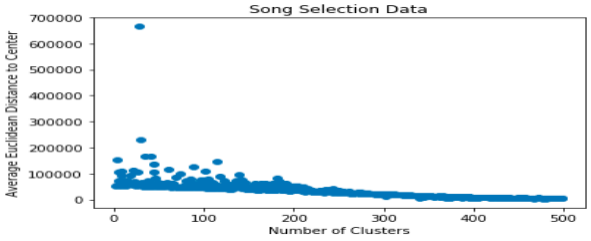
Data Set 2 (College Admissions):



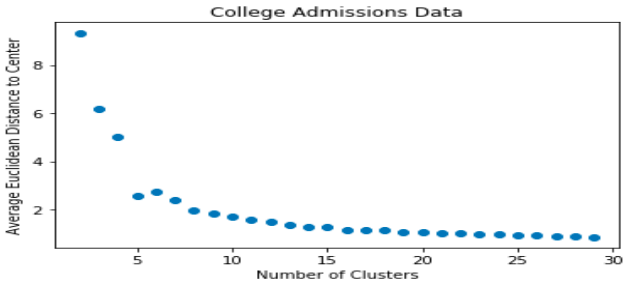
Analysis:

First, I used the k-means model simply to solve a classification problem. This is not the traditional approach to unsupervised learning because for most unsupervised learning problems, one does not even know what classifications the data may fall under. However, I thought the results were very interesting. I set the k value of the model at 2 for both data sets, knowing there were two possible classifications for both data sets. The above figures show points graphed for each data set. It is important to note that Data Set 1 has 10 features and Data Set 2 has 3. I could not graph in n-planes, so I simply chose two features for each data set and graphed the points based off of them. For both data sets, the left graph shows the points divided by their actual classifications and the right shows what k-means returns. It is interesting to note that the algorithm identifies all fringe points correctly, but struggles with ones in the center. The scikit learn library has a classification report that measures how well k-means classification compares to the actual target values when taking into account that the algorithm does not map to specific categories. The classification report returned an average precision of 0.84 for Data Set 2 and 0.63 for data set 1. This shows two things. First, 0.84 is a spectacular score. It is even better than the accuracy of some of the neural networks from Assignment 1. This shows that K-Means works extraordinarily well for small data sets with a small number of features. This is due to the simple mathematical nature of the algorithm that allows it to simply average points to find optimal centers. In addition to this, a smaller number of features means the algorithm works in fewer dimensions, which makes calculating a center easier and more accurate. The small size of data set 2 also means fewer outliers that might negatively affect the centers and a more homogeneous data set that works perfectly for this kind of algorithm.

Data Set 1:



Data Set 2:



Analysis

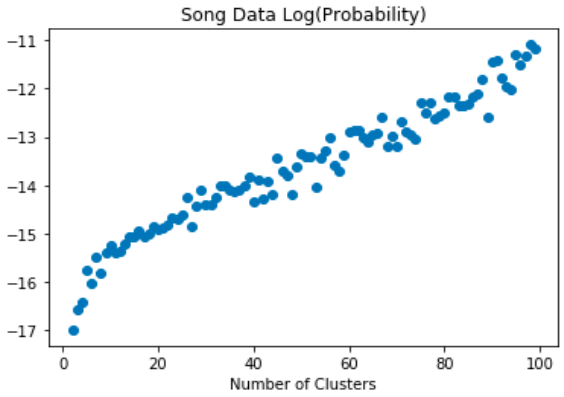
The above graph shows how the average Euclidean Distance of each point from its designated cluster's center varies depending on k, the number of clusters. Obviously, as k increases, the average Euclidean Distance will decrease. More clusters mean more centers which in turn means the likelihood of a point finding a close by center increases. Essentially, if k is high enough such that each point is its own cluster, the average Euclidean Distance will be zero. This situation is essentially the same as overfitting. The algorithm has studied the data set so much that it is essentially just a copy of the data set and therefore useless. So how does one find the ideal k such that the algorithm does not over fit, but at the same time has a high enough k to expose interesting insights regarding the data. The most common way of determining an optimal k is using the elbow method. If the data looks like an arm, the optimal k corresponds to the "elbow." This elbow corresponds to the point directly before the data begins to converge. This is after the data has decreased significantly and is starting to asymptotically. For Data Set 2, the optimal k would be 4. For Data Set 1, it is clear that running k-means does not follow the same pattern. This is due to the fact that Data Set 1 has 10 features. This means that the algorithm is operating in 10 dimensions. It is very difficult to find an appropriate center for all points between iterations when averaging the points in 10 dimensions. It is possible that only 2 or 3 of the features are relevant in classifying the data for a certain point, but k-means will examine all 10 and attempt to average across all dimensions, causing a lot of noise. This is apparent in the fact that the Average Euclidean Distance varies greatly for points that are within 1 or 2 "ks" of each other. This should not happen, but when one induces randomness and attempts to reevaluate across 10 dimensions, there will be variation between similar k values. For data set 1, it is not possible to determine an optimal k because the graphs do not converge and the average Euclidean distances do not vary much between different clusters. Perhaps there will be improvement after running dimensionality reduction on the data set.

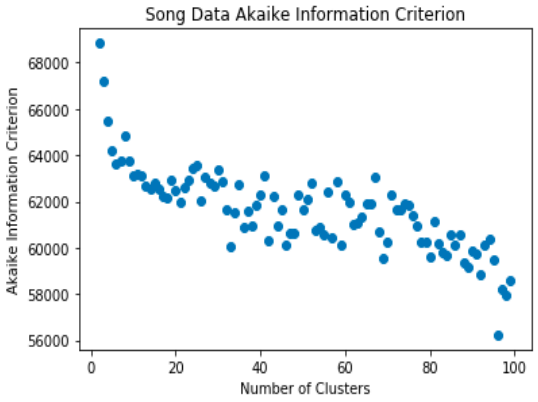
Expectation Maximization

Introduction: K-Means clustering does clustering by definitively assigning each point to a specific cluster based on which cluster's centroid the point is closest to. There are several issues with this type of clustering. For one thing, if a point were equidistant to two clusters, the cluster it belonged to would be determined entirely arbitrarily. In addition to this, the algorithm does not encode a lot of valuable information regarding the data. Suppose you are given a data set and a k value of 2. One point is of Euclidean distance 1 from cluster A's centroid and 1000 from cluster B's centroid. Another is of Euclidean Distance 500 from cluster A's centroid and 501 from cluster B's centroid. These two datum are very different. One is almost certainly n member of Cluster A, but the other could belong to either. K-Means clustering treats these two points entirely identically and provides no means of encoding their differences. Expectation Maximization provides a solution to these problems. Essentially, it creates a probability distribution for each cluster that gives the likelihood for each point that the point belongs to the given cluster. The algorithm starts by creating k Gaussians randomly. Then it assigns each point the probability of it belonging to each of the k Gaussians. Then the k Gaussians use the points assigned to them to recalculate their means and variances. This process is repeated until the means and variances converge.

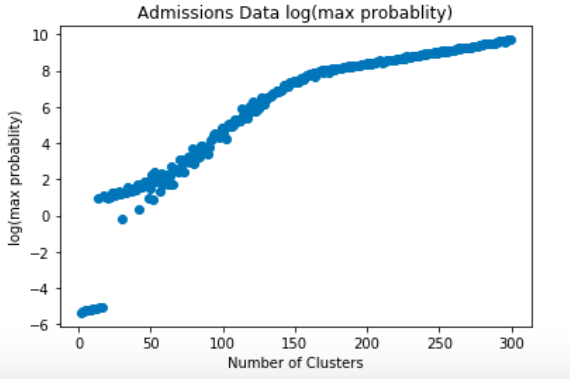
Since Expectation Maximization is composed of probability distributions rather than concrete clusters and Euclidean distances, we will have to use a different method from k-means to determine the optimal number of clusters. We will look at the ln(M), where M is the maximum probability and the Akaike Information Criterion, or $2k - 2\ln(M)$, where M is the maximum probability and k is the number of features as a function of the number of clusters. The ln(M) is something that we want to maximize to a certain degree. A high ln(M) means that the model is well conditioned to the data and confident regarding the data. A very high ln(M) evidences the model is to conditioned to the data, or overfitted. For example, ln(M) will be highest when M is 1. M will be at or near 1 when k equals the number of data points, something we want to avoid. At the same time, we want a high ln(M) so that the model is confident in the data. Therefore, we will use the "elbow method" to evaluate the optimal number of clusters and look at the point before the data starts behaving asymptotically. A lower AIC score corresponds to a higher ln(M) and a lower number of features. Generally, a lower AIC score means a better model, but a very low AIC scored evidences overfitting. A high number of parameters (k) generally means the data is overfitted. That is why the AIC equation includes 2k. We will examine the point before the AIC graphs began behaving asymptotically to determine the optimal number of clusters.

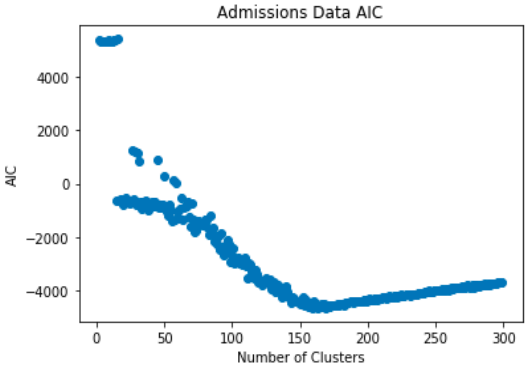
Data Set 1:





Data Set 2:





Analysis: For both data sets, the elbow appears at roughly the same point for the AIC graph and the log(max probability) graph. This is likely due to the fact that the AIC is the inverse of the log(max probability) adjusted based on the number of features to encode overfitting in the score. The elbow represents the point at which the model has reached appropriately high probability levels without overfitting. For Song Data, if you disregard some noise, this occurs around k=10. For Admissions Data, this occurs at a much higher k=150. It is interesting that Data Set 1 actually reached a convergence point with AIC, as it did not reach one with K-Means. This is due to the fact that data with a large number of features are harder to hard classify into a specific category. There are so many variables for each data point, that the data works much better with soft clustering, where it may be shared by several clusters. It is also noteworthy that the Admissions Data required a much larger k to converge. Even then, it did not converge smoothly. This shows that k-Means clustering is better for data sets with a smaller number of features. This is because data sets with fewer features can easily be separated into definitive groups based on their features, while data sets with a large number of features must "toe the line" from group to group because their massive amount of information does not allow them to fit perfectly into one group or another.

Conclusion: In conclusion the optimal number of clusters for data sets 1 and 2 respectively in K-Means clustering are 2 and 4. For Expectation Maximization, the respective optimal number of clusters are 10 and 150. This shows that Expectation Maximization, in general, performs better with more clusters. Also, Expectation Maximization seems to perform more optimally with higher dimensional points, while K-Means seems to do better with points that have fewer features.

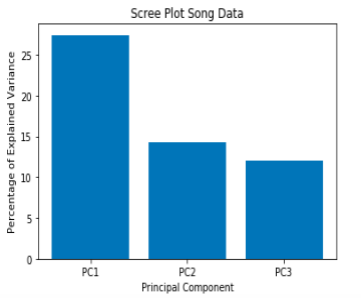
Dimensionality Reduction

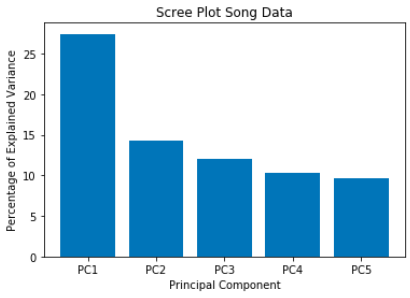
Introduction: Richard E. Bellman coined the term "The Curse of Dimensionality." In statistics and data analytics, this refers to the multitude of problems that arise when analyzing and classifying data in high dimensional spaces. One problem is that to analyze data in very high dimensions and find statistically significant elements, one needs an incredibly large data set. In the realm of classification, algorithms seek to find similarity between data points. In extremely high-dimensional problems, it is much higher to find similar properties among data because each point spans several dimensions. We witnessed this problem when running K-Means on data set 1, a 10 dimensional data set. The classification report produced poor results and the graph of the optimal number of clusters would not converge. Dimensionality Reduction problems seek to fix this using both Feature Selection and Feature Transformation. Feature Selection algorithms determine which m-features of an n-dimensional data set are most significant, so that other algorithms may be run on the data set while only looking at those features. Feature Transformation algorithms use linear algebra techniques like combinations to turn n features into m features where generally m < n, while sacrificing minimal information. We will look at these four Feature Transformation algorithms: Principal Component Analysis, Independent Component Analysis, Randomized Projections, and Information Gain.

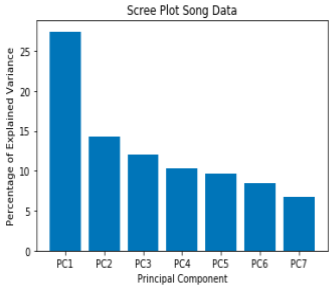
Principal Component Analysis

Introduction: Principal Components Analysis uses techniques from linear algebra to break down n-dimensional data to m dimensions (m < n). This is done by determining which directions projecting the data onto best maximize the variance of the data. This dimensions are called Principal Components and the data is projected onto these vectors. Here is how the algorithm works: The data is represented by an n x m matrix X. M is the dimensionality of the data and n is the number of samples. When one takes the Eigen Decomposition of the m x m matrix X^TX, one gets m eigenvalues (one for each dimension) and a matrix W made up of m eigenvectors corresponding to each eigenvalue. The eigenvalue is akin to the variance of the data when projected onto the direction of its corresponding eigenvectors. The algorithm finds the r (new number of dimensions) vectors corresponding to the r highest eigenvalues. These r vectors are the principal components and the data will be projected into the directions of each of these principal components, thereby lowering the dimension while maintaining almost all information. When you put the r vectors into a matrix you have W_r, a m x r matrix. Multiplying X by W creates the transformed data set, a n x r matrix. We will perform Principal Component Analysis and examine how much variance of the data set each Principal Component accounts for. Then we will run the clustering algorithms on the reduced data set and examine how the algorithms behave with the reduced data. Finally, we will run the neural networks from Assignment 1 on the data and examine how well they perform with fewer features.

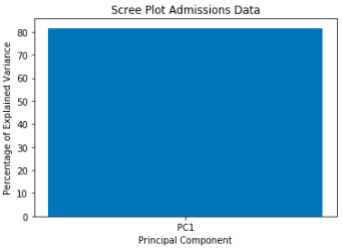
Data Set 1:

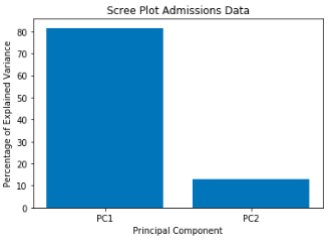


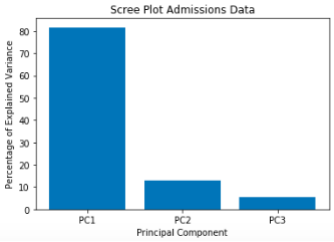




Data Set 2:





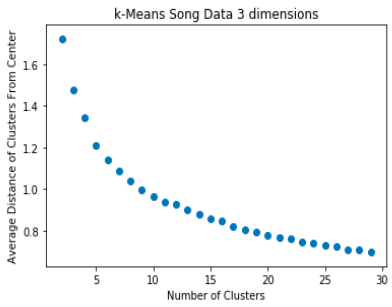


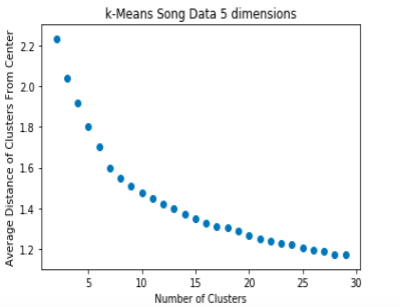
Analysis:

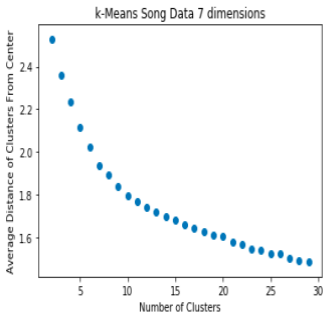
Above are three Scree plots for each data set. A scree plot maps each principal component of the PCA reduced data to the percentage variance of the data that the principal component captures. Data Set 1 has 10 dimensions and Data Set 2 has three. PCA was performed on both data sets, reducing the first set to 3, 5, and 7 dimensions and the second one to 1, 2, and 3 dimensions. It is important to note that between each iteration of the model for Data Set 1, the first three principal components account for the same percentage variation in the data. This makes logical sense as no matter how the algorithm is run, it will create m components and return the specified number. Therefore, changing the number of dimensions that you reduce the data to will not impact the way the algorithm is run aside from what components it returns. It is also impossible to note that the only graph that captures 100% of the variation in data across all components is the third graph in data set 2. This is because in that model the data was reduced from 3 dimensions to 3 dimensions. This means that in any case of reduction some variation in the data will be lost, so we want enough dimensions such that we maintain the majority of the variation, but not so many that we have the "Curse of Dimensionality" again. For Data Set 2, one component is sufficient to maintain 80% of the data's variation. For Data Set 1, 5 gives us 65% of the original data's variation, which seems sufficient. We will find what works best when testing PCA modified data on our Neural Network from Assignment 1.

K-Means Clustering

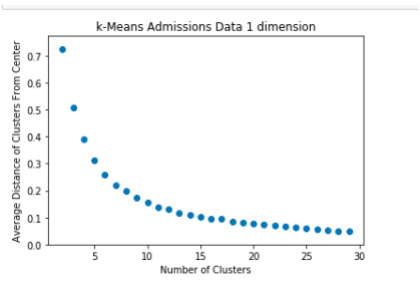
Data Set 1:

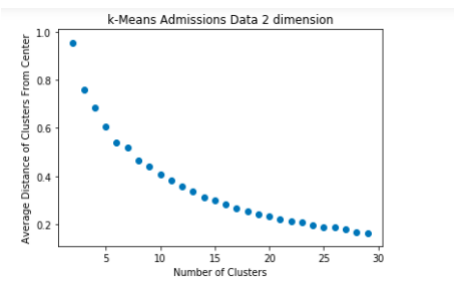


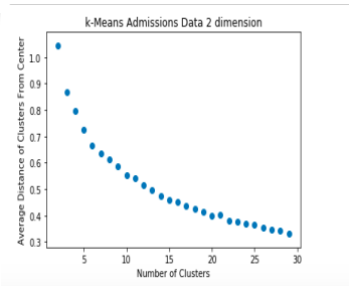




Data Set 2:





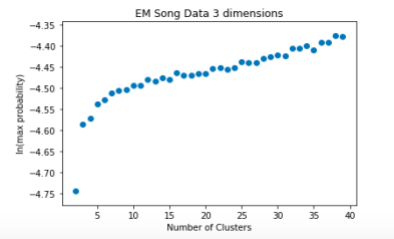


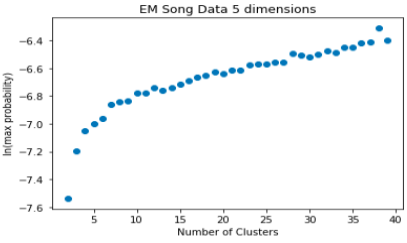
EM

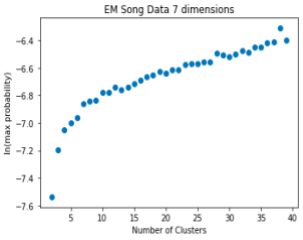
Analysis:

We performed dimensionality reduction on all both data sets. For each dimension, we performed K-Means Clustering, varying the number of clusters and for each model, determining the average Euclidean Distance of each data point to its cluster's center. We found that PCA radically changes how the K-Means clustering algorithm operates on Data Set 1 (originally had 10 dimensions. If one were to compare the graphs of K-Means on Data Set 1 after PCA to before, you would see that the data actually converges, making it possible to estimate, accurately, the optimal number of clusters. For each number of dimensions, it seems that the optimal number of clusters for Data Set 1 (using the "Elbow Method") is around 9, compared to and inconclusive value without PCA. This increase in optimal number of clusters is due to the fact that the graph actually behaves asymptotically on the reduced data. It did not do so when K-Means was performed on the unreduced data. Varying the number of dimensions does not affect the optimal number of clusters, but looking at the y-axis for each graph shows a clear increase in average Euclidean Distances as the dimensionality goes up. This means performing PCA more aggressively (decreasing the number of dimensions by more) results in more tightly bound clusters, while performing it less aggressively results in loosely bound clusters. A lower Euclidean distance is a sign of an accurate model, so more performing PCA more aggressively seems to be the optimal solution the Unsupervised Learning problems. As for Data Set 2, the optimal number of clusters seems to be around 7, compared to 4 without PCA. It seems that after performing Dimensionality Reduction the optimal number of clusters increases. This makes sense as fewer parameters allows for the algorithm to more easily "see" stark differences between data points.

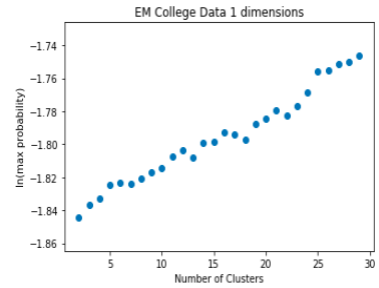
Data Set 1:

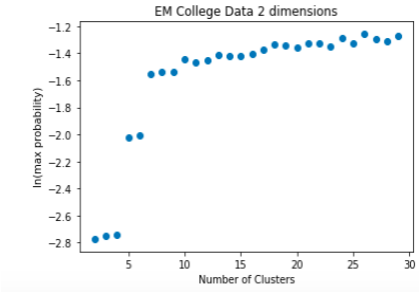


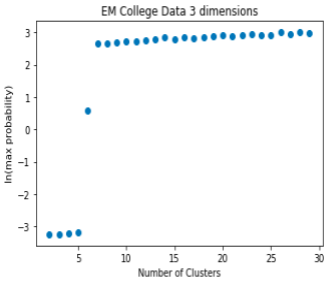




Data Set 2:







Analysis:

The above scatter plots graph ln(max probability), a measure of the accuracy of the Expectation Maximization algorithm, for each number of clusters for 3 different models of the EM algorithm for both data sets. Before the EM algorithm was run on either data set, PCA was run on both data sets reducing the data set to the number of dimensions specified above. For Data Set 1, before running EM, the dimensions of the 10 dimensional data set were reduced to 3, 5, and 7. For Data Set 2, the 3 dimensional data set was reduced to 1, 2, and 3 dimensions. For data set 1, using the elbow method, the optimal number of clusters was determined to be 9 for each dimension. For data set 2, it was 7 for 2 dimensional and 3 dimensional models. The 1 dimensional model did not converge. Even for the 2 and 3 dimensional models on data set 2, the graphs produced inconsistent results. This is due to the fact that soft clustering techniques do not seem to perform well with data sets that have a low number of dimensions. We saw this occur in the Expectation Maximization section as well. This is likely due to the fact that lower dimensional data sets are a better fit for "hard and fast" relationships with their clusters because there are fewer features to classify them based on. Also, both data sets showed a lower optimal k than they did in the EM section. In fact, for data set 2 the optimal k decreased from 150 to 7! This is likely due to the fact that a fewer number of features means that the data is more homogeneous. There are fewer features that separate points from one another. Therefore, fewer clusters are required for optimal performance as evidenced by the data.

Neural Networks

Data Set 1:

Song Data Neural Network (13 layers)			
3 Dimensions	5 Dimensions	7 Dimensions	Original Data Set
0.488448845	0.547854785	0.537953795	0.523337223

Data Set 2:

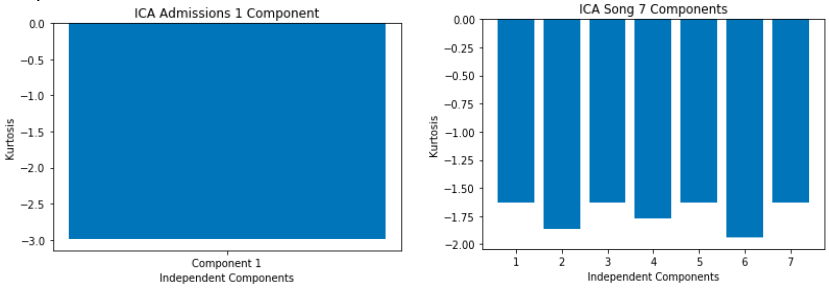
Admissions Data Neural Network (6 layers)			
1 Dimension	2 Dimesions	3 Dimensions	Original Data Set
0.453333333	0.586666667	0.52	0.778823529

Analysis: Unsurprisingly, PCA reduced the neural network accuracy for data set 2. This is because the College Admissions data set did not suffer from the "Curse of Dimensionality." The data only had three dimensions, making it perfect for a Neural Network with 6 layers. A small number of features prevented the neural network model from over fitting to the data and gave us a test score of 0.78 in Assignment 1. Reducing the number of features through linear combination only resulted in information loss and thereby caused poor scores. There was a slight increase in the testing scores for the Neural Network where PCA reduced the dimensions to 5 and 7, but the difference is only 0.01-0.02. This is surprising because a fewer number of features would prevent the model from overfitting and thereby create a much higher testing score. Perhaps the loss of information negated some of the benefits of dimensionality reduction.

Conclusion: PCA is a complex and effective algorithm for reducing the dimensionality of a data set using techniques form linear algebra. However, as evidenced by the Neural Network tests, there are likely some issues with preserving information from the original data set as it is transformed.

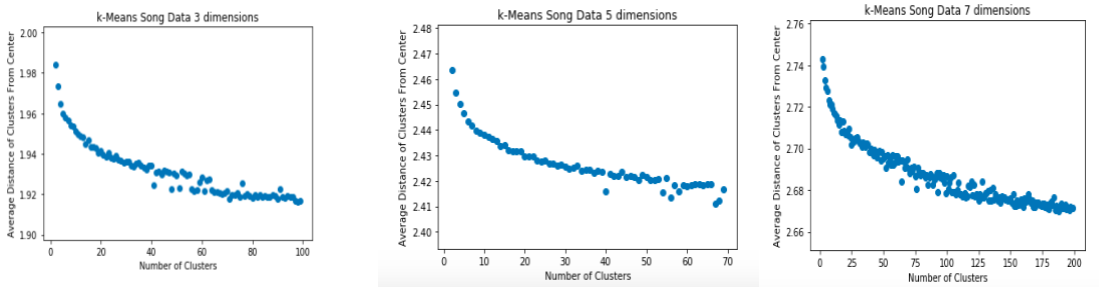
Independent Component Analysis

Introduction: While Principal Components Analysis seeks to create a new feature space to maximize variance across each new feature, Independent Components Analysis creates new features and seeks to maximize the statistical independence between each new feature and maximize the mutual information between the newly created features and the original features. This is based on the assumption that each feature of the original data is composed of some independent components and that isolating these components will create a better representation of the data. ICA seeks to create independent components. One way we can test how independent the distribution of each component is examining their kurtosis values. Kurtosis is a measure of how much of the data of a distribution is condensed in its tail ends, more specifically how much the tail ends' densities differ from a normal distribution in thickness. We are using the kurtosis measurement to see how much the components differ from a normal distribution. A normal distribution has a kurtosis of 3 and we subtract 3 from all our kurtosis calculations. A higher kurtosis means starker differences from a normal distribution and thereby implies mutual independence among components. Essentially, the farther the kurtosis value is from 3, the more independent the components and the better the algorithm's performance is.

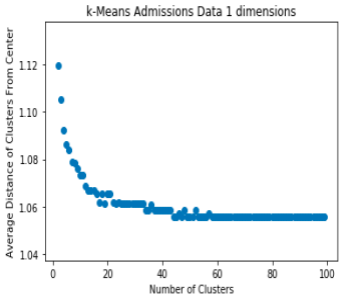


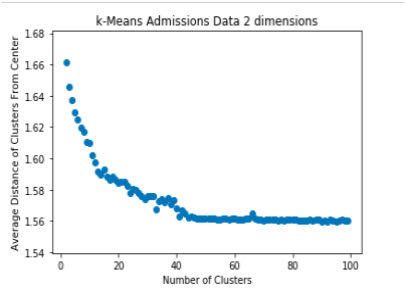
Analysis: Above are the kurtosis scores for the admissions data (reduced to 1 dimension) and the song data (reduced to 7 dimensions). I did not include graphs for lower dimension reductions because they did not differ much from the above graphs. None of the scores are near 0, so this implies that the algorithm did well to create independent components. It seems the algorithm was able to better separate the admissions data into independent components than the Song data, as evidenced by the higher kurtosis score. This could simply be because it easier to filter out noise for lower dimensional data. Now let's examine the K-Means clustering algorithms on the reduced data sets.

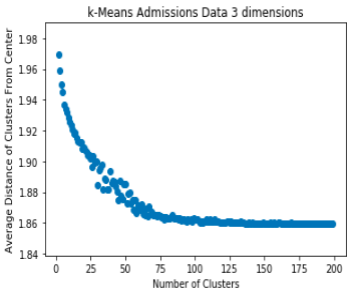
Data Set 1:



Data Set 2:

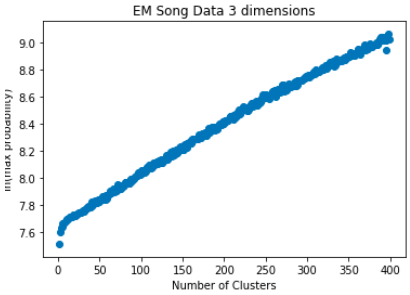




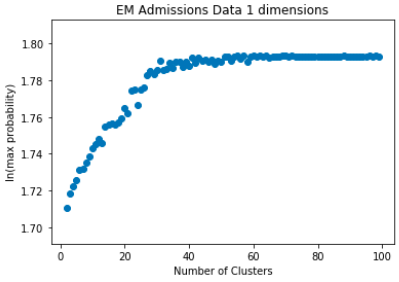


Analysis: After reducing data set 1 to 3, 5, and 7 dimensions and data set 2 to 1, 2, and 3 dimensions, I ran k-Means on each reduced data set for a variable number of clusters and graphed the corresponding average Euclidean distance of each point to its cluster's center. In the graphs above, the total number of clusters I built models off of varies for each data set. This is because the models took a different number of clusters to converge for each data. In fact, k-Means on data set 1 at 7 dimensions does not converge! Perhaps it would if we had extended the maximum number of clusters to run the model with to around 300, but time simply would not permit this. After running ICA and using the elbow method, the optimal number of clusters for data set 1 seems to be 20 and for data set 2 it seems to be around 19. This is compared to 9 (data set 1) and 7 (data set 2) from k-Means after PCA and "inconclusive" and 4 without dimensionality reduction. Essentially, running K-Means after ICA requires more clusters for the algorithm to behave optimally then running K-Means after PCA. This basically means that running ICA makes the data less homogeneous than running PCA. This makes intuitive sense as ICA seeks to maximize independence across components. For these data sets, it appears that PCA is more optimal because it is easier to determine the optimal number of clusters. Finally, if one looks at the y-axis of these graphs, they'll see that reducing the data points to lower dimensions causes lower Euclidean distances, and thereby more certainty in the groupings of the clusters. This shows hope for aggressively reducing dimensions. Now let's examine how Expectation Maximization behaves with the dimensionality reduced data.

Data Set 1:



Data Set 2:



Analysis: I only included the graphs clusters vs ln(max probability) for data sets 1 and 2 when the algorithms were run on the data sets reduced to 3 and 1 dimension(s) respectively. This is because the clusters vs ln(max probability) for higher dimensions added no new information other than the average ln(max probability) being lower when the data sets were reduced to a higher number of dimensions. This is due to the fabled "Curse of Dimensionality." A higher number of dimensions makes the data appear to have more disparity thereby decreasing how certain the algorithm can be that a data point belongs to a certain cluster. Again, using the "elbow method," the optimal number of clusters for data set 2 for EM after ICA was performed on the data appears to be around 35, compared to 7 for EM after PCA was performed and 150 without dimensionality reduction. It makes sense that ICA took longer to converge than PCA because ICA is designed to make the data less homogeneous, requiring more clusters. It seems that for data set 2, EM works a lot better after some dimensionality reduction is done. For data set 1, EM performs worse after dimensionality reduction is done, but k-Means does perform better after some sort of dimensionality reduction. Data Set 1 does not seem to converge at any point. Perhaps it converges with a much higher k-value, but time would not permit such testing. The explanation for this is that each component is so independent that the data becomes too unique and as a result, EM requires many clusters to behave optimally. Now let's examine how neural networks behave with ICA reduced data sets.

Data Set 1:

Song Data Neural Network (13 layers)			
3 Dimensions	5 Dimensions	7 Dimensions	Original Data Set
0.521452145	0.493333333	0.586666667	0.523337223

Data Set 2:

Admissions Data Neural Network (6 layers)			
1 Dimension	2 Dimesions	3 Dimensions	Original Data Set
0.506666667	0.493333333	0.493333333	0.778823529

Analysis: Above are charts describing the testing scores of neural networks, built with the optimal number of layers (determined from Assignment 1). The results are not all that different from the PCA results. The ICA reduced data sets caused a lowering of the testing score for all possible dimensions with Data Set 2. This is because data set 2 only had 3 dimensions to begin with. The tests show that

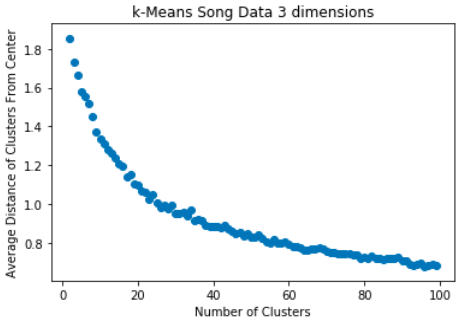
data sets with too many dimensions may cause a neural network to over fit to the data. Clearly, three dimensions is not too many. Lowering the dimensionality of the data set caused a loss of information, resulting in poorer test scores. For data set 1, running ICA on the data set and reducing it to 7 dimensions showed approximately 1% improvement in the testing score of the learner. This is not very significant, but the improvement could be due to ICA mitigating the "Curse of Dimensionality."

Conclusion: In general, running clustering algorithms on data sets after running ICA seems to yield much higher optimal numbers of clusters. This is undoubtedly due to the fact that ICA seeks to maximize independence across components, making data more statistically unique, thus K-Means and EM will require more clusters for optimal performance. ICA did not show significant improvement in Neural Networks.

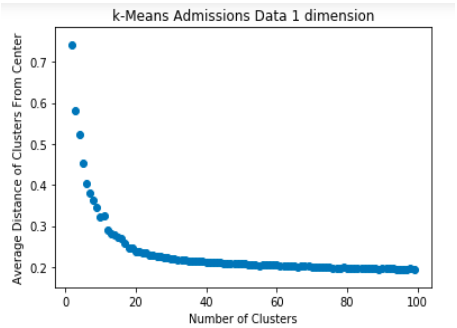
Randomized Projections

Introduction: Randomized Projections is another feature transformation algorithm that can be used to project data onto higher or lower dimensions using the technique of randomization. Essentially, the algorithm uses the new number of dimensions and generates a randomized matrix to project the data into a new feature space. This algorithm is unique because it can be used to project the data into higher and lower dimensions. Sometimes higher dimensional projections reveal useful information about the data set. Below are graphs of EM and K-Means run on the data sets after the data sets have been projected into a variety of dimension.

Data Set 1:

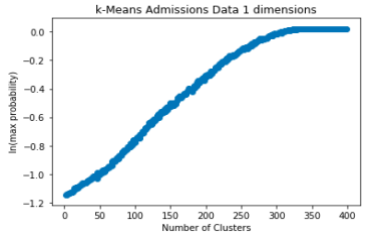


Data Set 2:

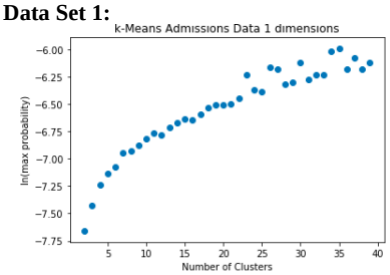


Analysis: Above I only included graphs for the data sets reduced to one dimension because the graphs of the data in other dimensions followed similar trends. Based on the "Elbow Test," the optimal k values are 12 and 17 for data sets 1 and 2 respectively. This is compared to "inconclusive" and 4 without dimensionality reduction, 9 and 7 with PCA, and 20 and 19 after ICA, so it takes longer for the data to converge with ICA but shorter with PCA, putting this somewhere in the middle. The curves here are much smoother and clearer than with any other algorithm. This is an indicator of the transformed data maintaining information from the original data. Now let's examine how EM behaves with this reduced data.

Data Set 2:



Data Set 1:



Analysis: Again using the elbow method, data set 1 converges around 25 clusters and data set 1 converges around 300 clusters (reversed above). This is compared to 10 and 50 without reduction, 7 and 9 with PCA, and 35 and inconclusive (possible infinitive) with ICA. Again, this algorithm seems to converge at a time between the other two. It is interesting that a one-dimensional data set takes so many clusters to behave optimally. A possible explanation is that so much information is lost in the reduction process that the data points lose all homogeneity, requiring more clusters. Now let's examine the neural networks.

Admissions Data Neural Network (6 layers)			
1 Dimension	2 Dimesions	3 Dimensions	Original Data Set
0.546666667	0.453333333	0.533333333	0.778823529
Song Data Neural Network (13 layers)			
3 Dimensions	5 Dimensions	7 Dimensions	Original Data Set
0.458745875	0.495049505	0.471947195	0.523337223

Analysis: There was no improvement with the testing scores of the neural networks after applying Randomized Projections. This is likely due to the fact that too much information is lost in the reduction process. If these data sets had more dimensions, perhaps these algorithms would result in more improvement.

Conclusion: In conclusion, with clustering, Randomized Projection appears to result in significant improvement in clustering in comparison to ICA. The graphs are much smoother and actually converge. However, PCA seemed to create the best results with clustering.

Neural Network

Introduction: Below are the accuracies of the neural networks when using clustering to reduce the dimensionality of the data sets.

Algorithm	PCA	ICA	RP	NONE
Data Set 1	0.56	0.51	0.62	0.523337223
Data Set 2	0.59	0.49	0.55	0.778823529

Conclusion: For the second data set all the dimensionality reduction algorithms resulted in a significant lowering of testing score. This is because the points of the second data set were only three-dimensional, thus not suffering from the "curse of dimensionality." Reducing their dimensions only resulted in a massive loss of information. For Data Set 1, all algorithms but RP produced no improvement. RP, however resulted in a 20% improvement in testing score. This makes sense because data set 1 had 10 dimensions. Reducing its dimensionality greatly improved over fitting, which occurs in neural networks that are given data with high dimensionality, this improving the testing score. PCA and ICA probably did not preserve as much of the data's information as RP did.

Overall Conclusion: After looking at 2 clustering algorithms and 3 dimensionality reduction algorithm, there is no clear winner. One important takeaway is that there is a clear winner depending on the data set. K-Means performs significantly better on data sets with a low number of features and not as well on Data Sets with a large number of features. This is because data with fewer features are more homogeneous, lending themselves to "hard" clustering. EM performs better on data sets with several features and poorly on data sets with a small number of features (in many cases data set 2 did not even converge with EM). This is because data points with more features are more diverse, lending themselves more to soft clustering. Of the three dimensionality reduction algorithms, Randomized Projections proved itself best at reducing data for Neural Networks. This is likely because RP's use of the random matrix conserves the most concrete information about the original data. While PCA only conserves variance and ICA only conserves independence, RP conserves a good combination of the two. PCA produced the lowest number of optimal clustering for both clustering algorithms across both data sets and ICA produced the highest. This makes sense because PCA preserves variance, the basis for clustering. ICA maximizes independence in projected data, which will ultimately make the data seem more diverse, requiring the clustering algorithms to pick up more clusters. Finally, all clustering algorithms seem to create the need for more centers with the low-dimensional data set 2. This is because when applied on data sets that do not suffer from the "curse of dimensionality," the algorithms can make the data seem more diverse, creating the need for more clusters. In the case of data set 2, the algorithms decreased the required number of centers because they made data set 2 seem more homogeneous. ICA seems inappropriate for clustering because it makes the data more diverse, but this may be necessary in some cases. Finally, the (mostly) poor results in the Neural Network can be attributed to information loss in algorithms that preserve variance and independence rather than essential, concrete information.