Jorge Betancourt

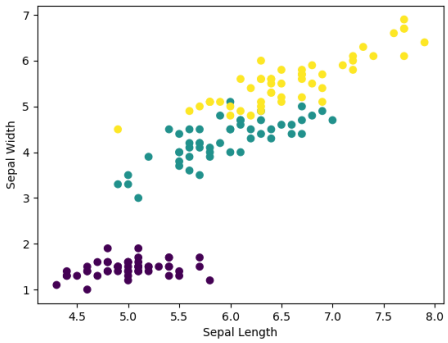
11/20/18

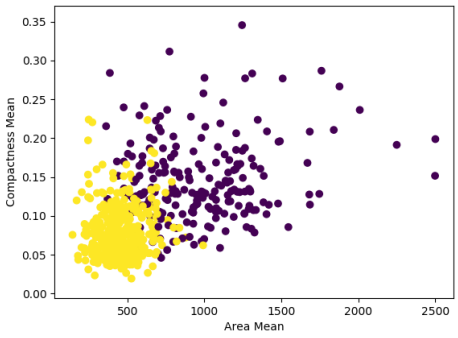
CS 4641

Assignment 3: Unsupervised Learning

Dataset Description and Intro

For this assignment, I am using the Wisconsin breast cancer dataset and Fisher's flower iris data set. I chose these datasets because I am familiar with the domain knowledge from my experience using them with other assignments. This way, I can properly explain my observations when using the unsupervised learning algorithms. Because they are also simple classification problems, I can also use their labels as a way to gauge the models' efficiency in properly clustering the data in a way we'd expect. Ideally, we'd expect the clustering algorithms to cluster data points into the different labels, each cluster representing a different label when k=|classes|. The breast cancer data set is more complex than the iris data set, with the iris data containing less features than the cancer diagnostic data. However, the iris data contains one more class than the cancer dataset, something that should effect run time in some capacity. Here are the ground truth clusters based on labels. To display the points, I chose 2 features that divide the data up rather well in a two-dimensional space.



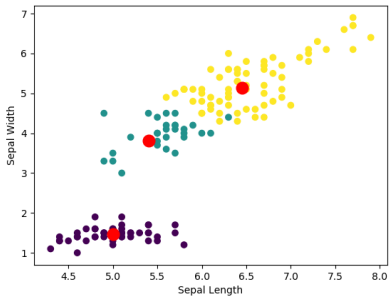


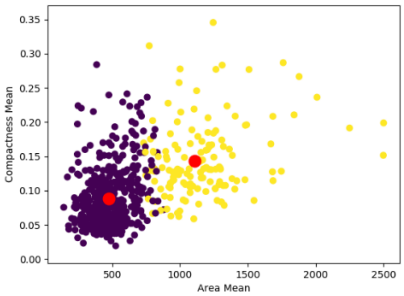
Unsupervised Learning Algorithms

K-Means

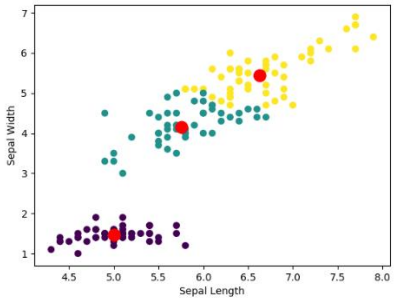
When running K-Means on both data sets, I took snapshots are different iterations after randomly selecting cluster centers. The results were much better than expected, and the clusters converged much more quickly than expected in terms of iterations. The cluster centers are shown in orange.

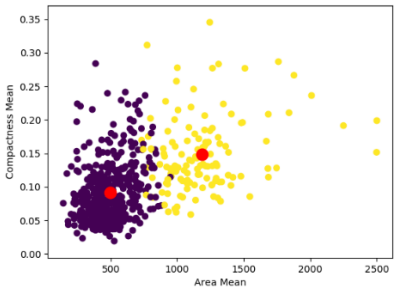
Iris | Cancer



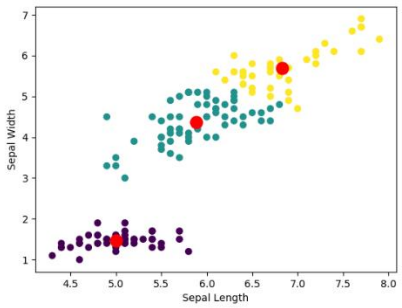


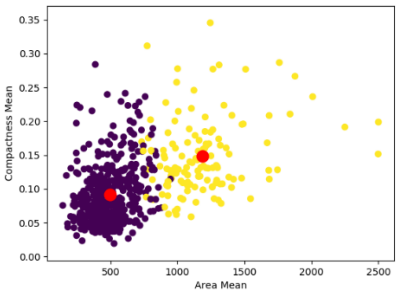
1 iteration





5 iterations



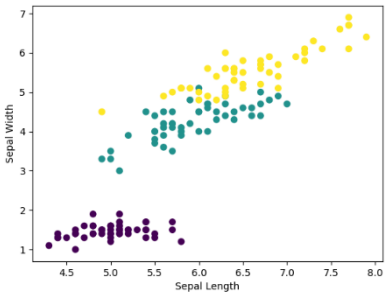


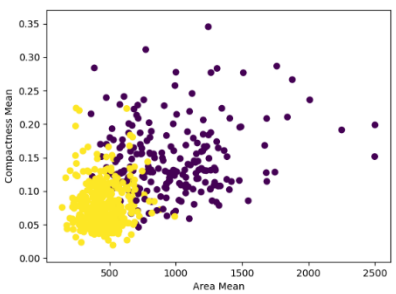
10 iterations

Something that should be considered is the majority of shifting in the data points closer to the boundaries of the clusters. While the shifting may not seem like a lot due to the minimal change in the orange points, it is important to remember that we are not visualizing most of the other features in the feature space, a problem that can be solved with PCA dimension reduction. Something interesting to note is the clustering algorithms ability to gain insight to the classes without touching label data. It was very close to the ground truth of clustering save a few errors where the data point could belong to either cluster. In this case, it would be helpful to see the probability of which cluster it is in, thankfully that is addressed with Gaussian Mixture Models.

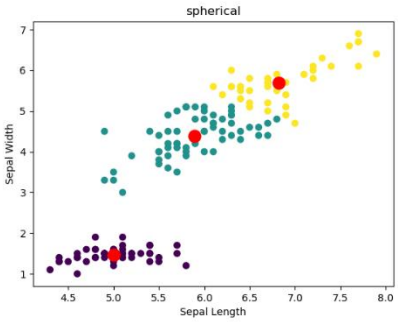
GMM

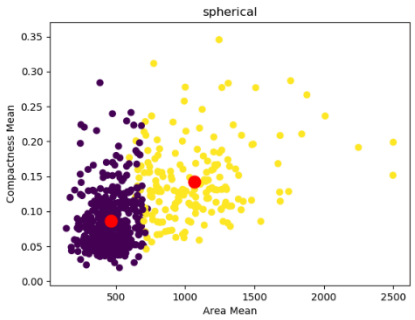
K-Means works well if the data points can be organized into spherical groups, but the downfall of K-Means comes when trying to cluster data into spherical groups when the data is clearly clustered in a different shape (like elliptical). GMM solves that by being able to fit different shapes of data by using different covariance parameters when maximizing the expectation that a certain point lies in the gaussian mixture. I expect to see similar results to K-means when using a spherical covariance because they both cluster similarly. I also expect to see more accurate results when using the diagonal covariance because a diagonal covariance will be able to consider outliers that are surrounded by another class f data points. GMM also provides a way to see each point's probability that it lies within a certain component. This is helpful for when assessing the clustering algorithms confidence. Here are the ground truths for reference:



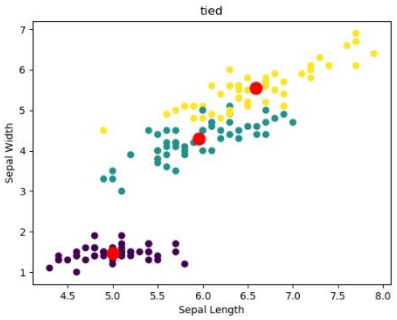


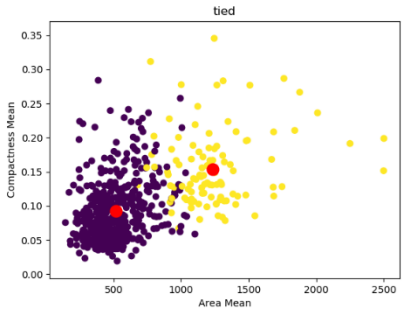
Iris Cancer



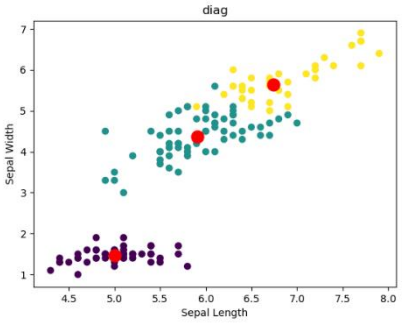


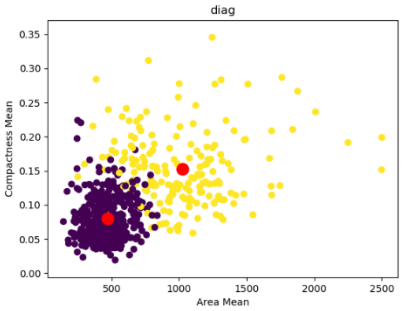
spherical



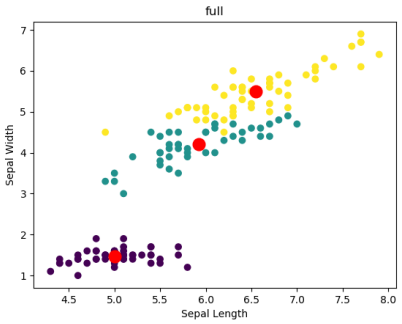


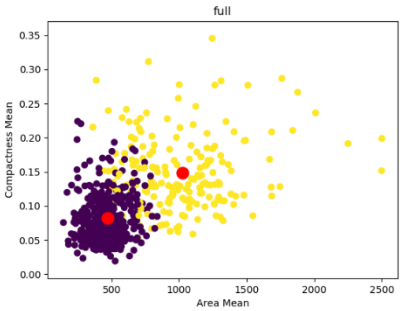
tied





diagonal





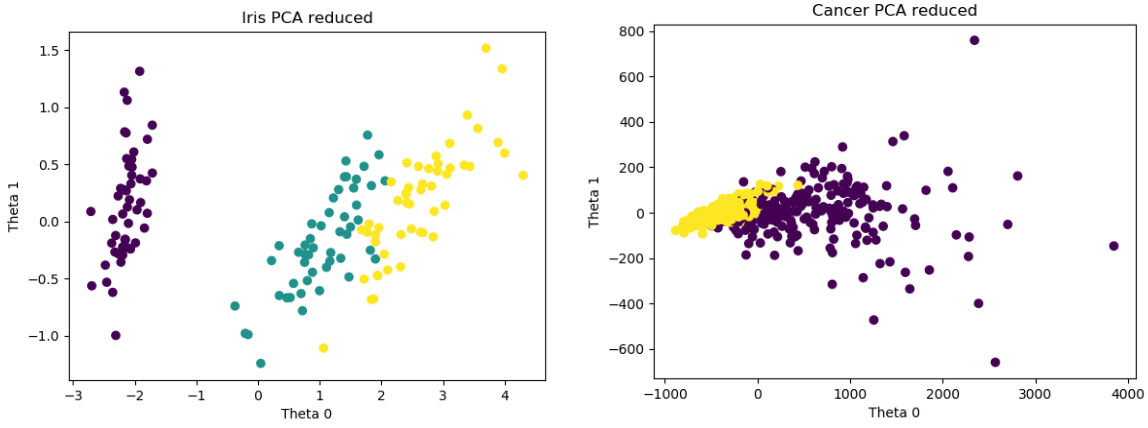
full

In the results above, we can see that there is a clearer divide between the clusters when using a spherical covariance for our components. This is similar to the clear divide in K-Means when clusters are determined by a radial distance from the cluster mean. We see more interesting results when the gaussian components are allowed to stretch and bend to better cluster the data.

PCA

So far, we have used two random features when visualizing the feature space. However, this causes a lot of information to be lost when graphing the data. To combat this, as well as the issue of dimensionality, we can apply PCA to transform our feature set. In order to visualize the data properly as well as test how well the data will be retained when strained, I will reduce the domain to two. This means the cancer data set will go from 32 features to 2 and the iris data set will be reduced from 4 to 2.

Ground truth (visualized with PCA):



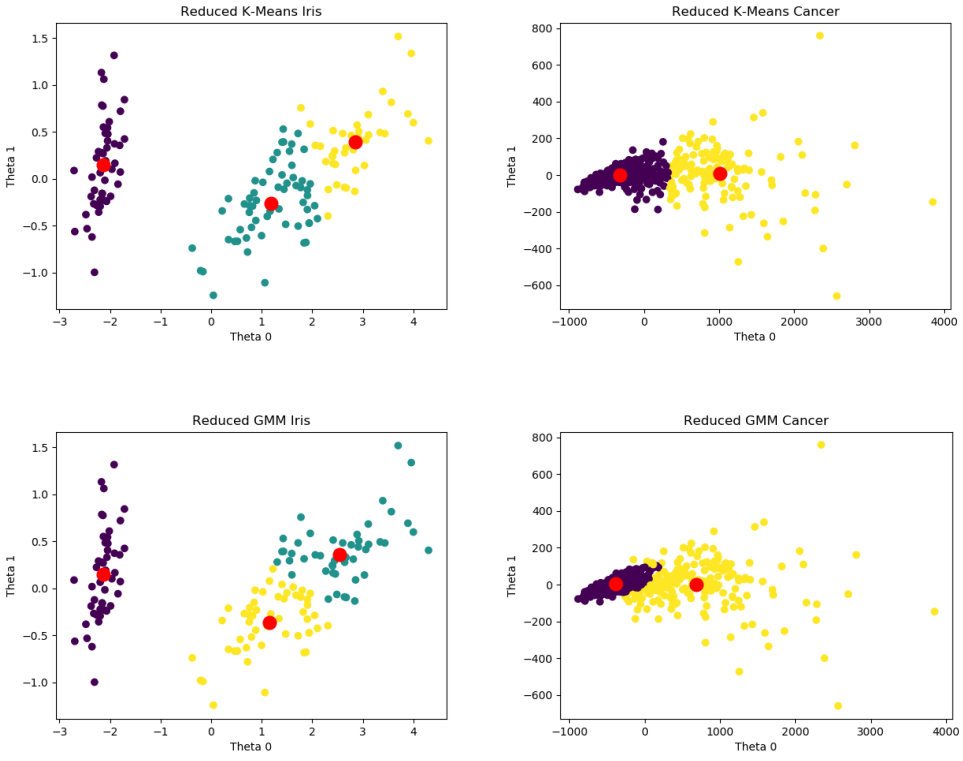
We see that when compared to the makeshift dimension reduction (where I simply selected 2 features) I used up until now, we see similarities in the spread. For example, iris data still has one isolated group with the other two groups close together. Also, we can see in the

explained variance that there was more information lost with the breast cancer data than with the iris data. This is because while with the iris data we went from 4 features to 2, the cancer dataset went from 32 to 2.

Reduced Dimension Clustering

K-Means vs GMM

Having domain knowledge of this classification problem is ideal, because we can use it to see which clustering algorithm would work best. Due to the shape of the clusters, circular based clusters may not be an ideal fit for this data. We can see this when we compare K-Means vs GMM with full covariances for each component.

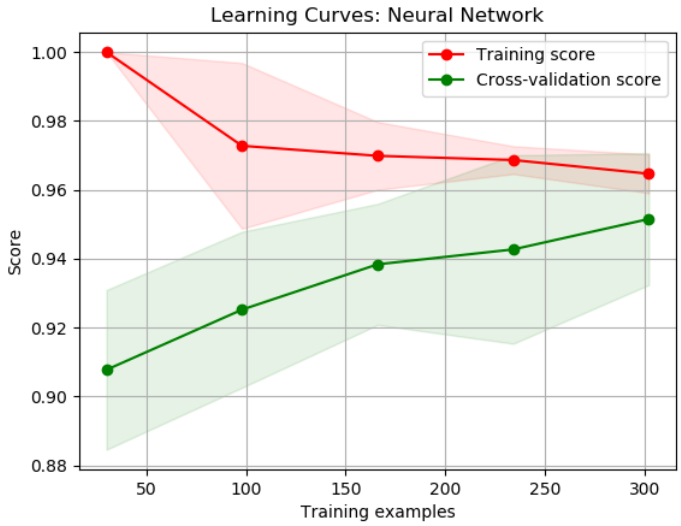


K-Means performed poorly in clustering the data-points based on class due to the non-spherical nature of the data. However, when using GMM it performed much better, especially with the cancer dataset, which looks almost exactly like the ground truth for the labels.

Neural Nets and Dimension Reduction

Reducing Feature Vector with PCA

I used the breast cancer dataset in assignment 1, so I'm using is again for this experiment. This time, during PCA, I specified an amount of information I was willing to lose and saw how many components I was left with. After specifying I wanted to retain .999% of my data, I was left with 4 features. The drastic change in dimensionality is most likely due to the correlation of the features. The network I ran contained the same optimized parameters as the network I worked with during assignment one for the sake of comparison.

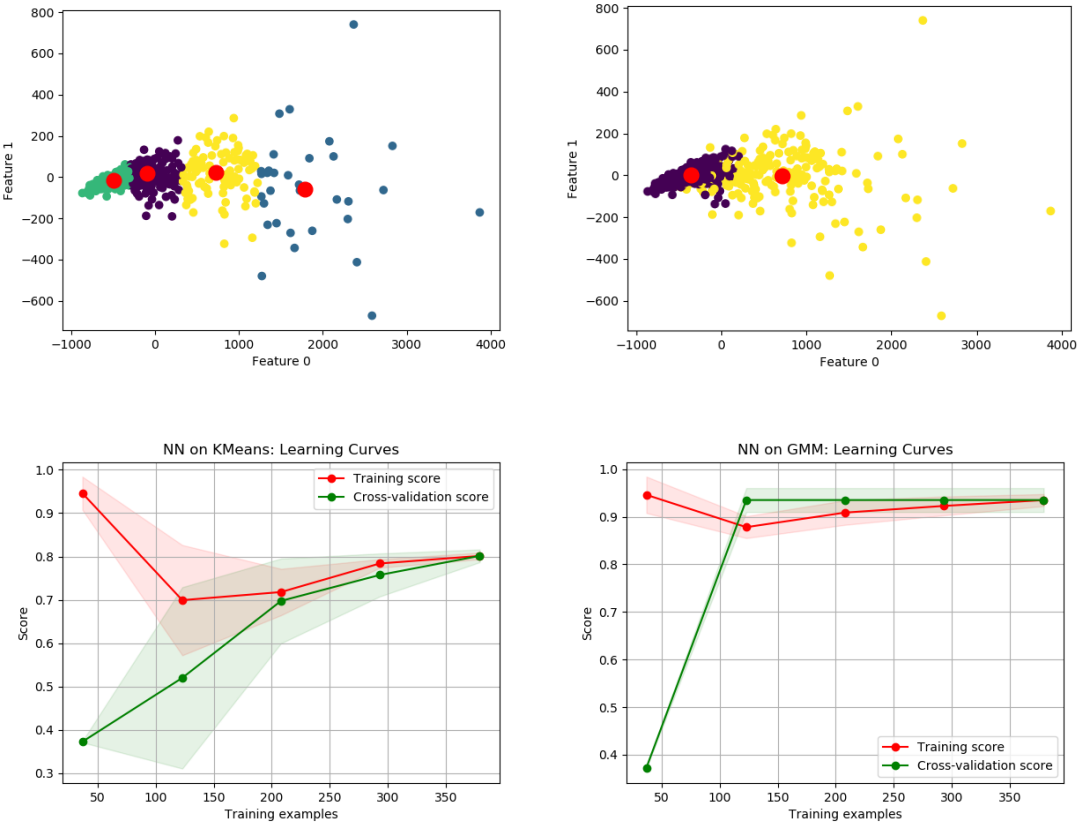


The neural network running on the reduced dimension breast cancer data was notably faster. As for accuracy, it performed just as well. This is probably due to the amount of information I wanted to retain during PCA.

Using Reduction as Features

Now we will see the results of using the clusters themselves as feature vectors in a neural net. For K-Means, I increased the amount of clusters and trained the network on a 1-dimensional feature space where the feature is the cluster label. For GMM, I took the probabilities of the data point's assignment and used those vectors as the input.

Note: While I clustered based on raw data, I transformed it into a two-dimensional space using PCA at the end for the sake of visualizing the data.



The above figure is how the clustering algorithm sorted the data, the bottom row is the features extracted from the sorting being used to train a neural network. The most obvious thing is how impressive GMM's dimensionality reduction worked on the neural network. This makes sense as this clustering is most similar to the ground truth grouping for the labels. K-Means

doesn't group the data into classes as well as GMM does, therefore it suffers in performance when trying to classify data points in between two cluster centers.

Conclusion

Something that I've learned over the course of this assignment is just how vital and effective dimensionality reduction can be. I have always assumed that more data leads to better results and that one can simply throw everything into a neural net and hope it learns the weights over time. I realized that while having many instances is always better, having an abundance of features does not necessarily improve the performance of your model. There are ways to improve clock speed, computing power efficiency, and sometimes even accuracy if one takes the time to analyze their data beforehand. That being said, reducing dimensionality too much can lead to important features being lost, so it needs to be used intelligently.

If I was to run more experiments, I would take the time to see the effects of PCA on extremely large datasets. I'd like to have meaningful results when comparing the wall clock time of pre-processed data versus the raw data. Experimenting with how much I can reduce the dimensionality before hitting a certain accuracy with a neural net would be interesting as well, unfortunately such experiments require more computing power and time to run.