Griffin Reichert
Clustering Project Report
CSE 347
3/21/2021

Introduction

In this project, I clustered the Cho and Iyer gene sequences datasets with k-means and spectral clustering to examine similar gene expressions. I was able to learn more about the benefits and consequences of each algorithm, along with some of the challenges faced when applying these clustering algorithms to real world problems

Description of Algorithms

I created two files to organize the clustering methods within python classes so that they could easily be reused for the different datasets. These can be found in `*/src*`. Before implementing the algorithms, I did some necessary data preprocessing. This included removing the outlier values which were denoted as -1 in the ground truth column and splitting the ground truth column into its own array, while keeping all of the data attributes in their own array.

K-Means:

The k-means algorithm is one of the most popular clustering algorithms because of its relative simplicity. In my `*kmeans.py*` file, I created a class to perform k-means clustering. In the constructor, this class creates all of the datatypes it will need. The important attributes for k-means are clusters and centroids. I used a list to store k-lists, where each of the k-lists stores the indices of each data point in that cluster. The centroids for each cluster were stored in a list, with the value of each centroid being a vector that is the mean of the features of each data point in the cluster.

To run k-means, the algorithm begins by randomly choosing k data points to be the centroids. Then, the algorithm does the following loop until we reach the max number of iterations, or the centroids converge:

1. It iterates over all data points, and for each point:
    a. Calculate the distance from the current point to each centroid
    b. Assign index of the current point to the cluster of the nearest centroid

2. Update the centroid values by taking the mean of all points in the cluster
3. Check if the new centroid values have not changed from the previous iteration. If this is true, then we say that the algorithm has converged.

Once the iterations stop, we return the labels in a single list, where the index is the index of the data point, and the value in that position is the cluster that point belongs to.

K-means is sensitive to initialization, so in order to achieve the best results, the random points chosen as the first clusters may need to be varied in order to find the best solution. For this implementation, that was done by using np.random.seed(x).

Spectral Clustering:

Spectral clustering is a more complex way of clustering than k-means, as it relies on a graph representation of the data and its eigenvectors to group similar points. To run spectral clustering, we must first create a Laplacian matrix to represent the data in a graph. I used the rule that the Laplacian can be found by subtracting the Adjacency matrix from the Degree matrix. To create the Laplacian, I first created a distance matrix (of size n-by-n) that holds the distances between each point. From the distance matrix, I was able to create the Adjacency matrix by placing a 1 at each position in the distance matrix where the distance between two points was below a certain threshold. This threshold was a hyperparameter that ideally would be optimized, however in my implementation I tried a variety and just picked the one that worked best.

After creating the Adjacency matrix, I was able to find the Degree matrix by summing up the number of other points that a point was connected to in the Adjacency matrix. The Laplacian was created by subtracting the Adjacency matrix from the Degree matrix (L = D – A). The Laplacian matrix allows us to do spectral clustering because it is symmetric, so it's eigenvalues will all be positive. If the graph is fully connected, there will be one eigenvalue of zero. If the graph is broken into multiple connected components, there will be that many eigenvalues of zero. Using numpy's built in eig function, the algorithm then computes the eigenvalues and eigenvectors of the Laplacian matrix.

These eigevectors are the core of spectral clustering. By identifying the position of the maximum difference between two eigenvalues, the number of clusters k can be found. Using this k-value, the k-eigenvectors corresponding to the k-smallest nonzero eigenvalues can be clustered using the k-means algorithm described above. This will result in the classification of k classes.

Performance of Algorithms

In an unsupervised data mining task such as clustering, having a variety of ways to analyze the performance of models is important. This project utilized both internal and external indices to compare the performance of the two algorithms on both datasets.

The internal index used was the within cluster sum of squares (WSS), which measures the cohesion of data points within a cluster. A lower WSS is typically better than a higher one since it means that the points in the cluster are closer together. That being said, a WSS of 0 is unrealistic and for interesting classification problems there will always be some difference between the points in a cluster. The following tables show the WSS for the Cho and Iyer datasets under the two models. The predicted WSS was based on the labels that my model predicted, while the Sklearn WSS was found using the labels from Sklearn's models. These were included to show how my algorithms performed against the standard models that exist in the professional world. The Truth WSS refers to the WSS calculated from the ground truth values that were provided in the dataset.

Cho Dataset Internal Index (WSS)

	Predicted WSS	Truth WSS	Sklearn WSS
K-Means	980.302	1244.001	1061.417
Spectral	1094.334	1244.001	1143.117

For the Cho Dataset, both my model and the Sklearn had a lower WSS when using k-means than spectral. My model also had a lower WSS than the Sklearn model, which could be because of a lucky random initialization of the points. The lower WSS indicates that the points in my clusters may have been closer together. This would also mean that the between cluster sum of squares (BSS) is higher because WSS + BSS is a constant. A higher BSS means that the clusters are more separated from each other which is typically a good thing.

Iyer Dataset Internal Index (WSS)

	Predicted WSS	Truth WSS	Sklearn WSS
K-Means	2682.826	14521.816	13329.143
Spectral	15162.455	14521.816	18821.219

The Iyer dataset proved to be a bit more complicated than the Cho dataset. For k-means the trends in WSS were similar to the WSS in the Cho dataset with my WSS being lower than Sklearn and the ground truth values. However, the spectral clustering did not go as well for the Iyer dataset. My predicted WSS along with Sklearn's WSS increased from the ground truth values, indicating a decrease in performance.

An import thing to note at this point, is that Sklearn output a warning when running spectral clustering on the Iyer dataset, indicating that "the graph was not fully connected, and spectral embedding may not work as expected". I was able to make my own spectral clustering work by using a very high threshold in creating the adjacency matrix. However, the Sklearn spectral results may not be reliable without further investigation and data preprocessing.

Moving from the internal index of WSS to an external index allows us to compare the performance of the models to the provided ground truth values. The external index used in this project was purity. Purity is a measure of the extent to which clusters contain a single class. A purity closer to 1 is ideal, as it indicates all clusters contain the correct classes.

Cho Dataset External Index (Purity)

	Predicted Purity	Sklearn Purity
K-Means	0.67098	0.67358
Spectral	0.52332	0.63472

For the Cho dataset, the Sklearn model has a higher purity than mine for both the k-means and spectral algorithms. This shows that it is a more accurate classifier. There are two interesting things to note. First, the purity of my predicted classes was only off from Sklearn by < 0.003, indicating their performance was very similar. Also, the purity decreased for both models doing spectral clustering, indicating that k-means was a better way of classifying the Cho dataset.

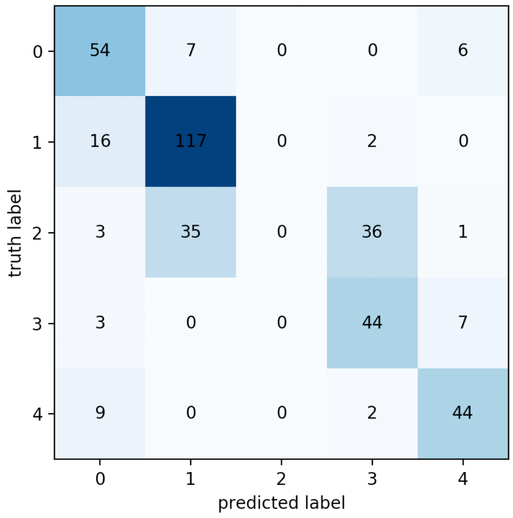
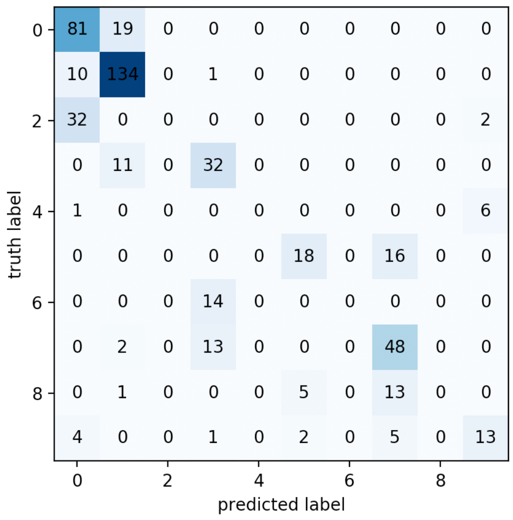
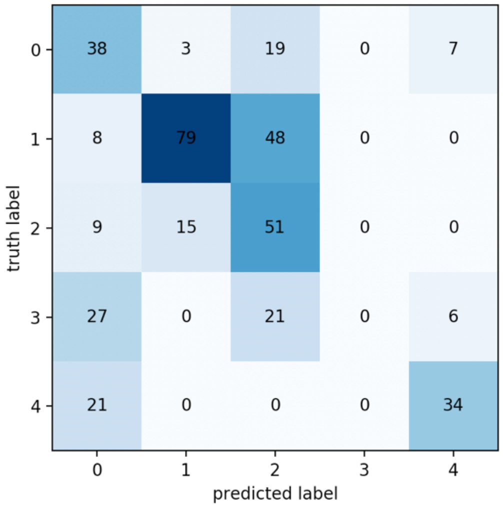
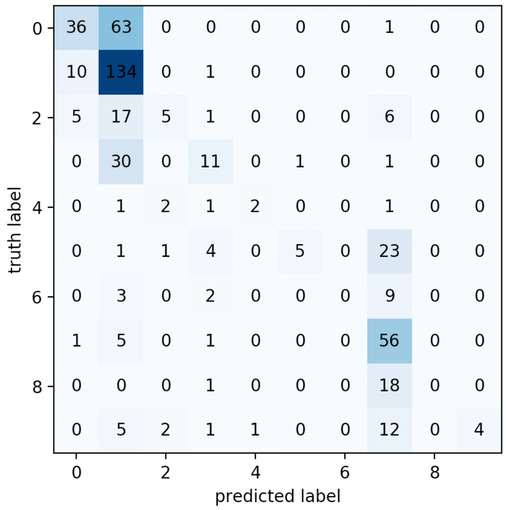
Iyer Dataset External Index (Purity)

	Predicted Purity	Sklearn Purity
K-Means	0.67355	0.54725
Spectral	0.52273	0.30165

The Iyer dataset was complicated to deal with in relation to purity. As shown in the warning discussed above, the Sklearn model needs some special preprocessing to accurately work with the Iyer dataset. However, my model had a relatively strong purity for k-means and spectral clustering. Again, we can see that the performance dropped from k-means to spectral clustering, as we saw with the Cho dataset.

One final way to visualize the performance of these clustering methods is through confusion matrices. These show graphically what the true labels were, and what my models predicted. A perfect confusion matrix would have all points on the diagonal.

Confusion Matrices

	Cho Dataset	Iyer Dataset
K-Means		
Spectral Clustering		

Conclusion

Overall, this project was an interesting way of applying clustering to a real-world problem of gene classification. I am proud that my models were able to attain similar performance to Sklearn’s models. Although the accuracy was only 67% at best, being able to replicate the standard models’ performance shows the merit of my implementation.

Ultimately, I think that special preprocessing of the data will be necessary to achieve better performance and accuracy. I think that it would be interesting to apply dimensionality reduction techniques such as PCA to the data. This could offer a performance improvement. Without this, it makes sense that these methods could not do better than 67% accuracy. Gene expression is a difficult problem, and if it was this easy to find solutions to these problems, the medical field would have done that already and applied these findings to cancer research and other cures. Pushing the boundaries of how we can use data to make decisions will continue to drive society.