Assignment 4

James Lowrey

In this lab K-means and DBSCAN were used alongside the Euclidean and Cosine distance metrics to process the reduced Reuters data. These clustering algorithms were tested with different parameters to attempt to produce the “best” clustering of the data. Cosine distance metric was found to provide clusters with less error, less cluster size deviation, and greater information gain. This matches other research group findings as Cosine distance typically works better on sparse datasets (Source 2). K-Means was also found to be a better algorithm. DBSCAN was expected to do better due to its non-globular nature and resistance to noise, but the results seem to indicate that the data was more globular in nature with similar density throughout.

Since the previous lab, the feature selection methodology was tweaked to improve clustering results. In the last lab, feature vectors were created from words with the highest TF-IDF values in the reduced (stop-words, numbers removed) corpus. The resulting vector has highly valuable words, but is extremely sparse. As text vectors are classically already very sparse, this reduction schema made the classifiers of lab2 highly inaccurate (~24% prediction accuracy). For this assignment, words near the median TF-IDF value were selected. This ensures that the words are meaningful (not noise) while producing feature vectors that are much less sparse. Rerunning lab2 with the new methodology resulted in an 160% increase in prediction accuracy (~40%).

Representative sampling was used to reduce memory usage and decrease clustering time. Sampling allows the number of data points processed to be much smaller than the original corpus and to maintain results similar to what would be expected if clustering was ran on the entire database. In this lab, the data size was reduced to a little less than ¼, a final size of 5000 samples. First the data was partitioned into 20 sections, and the same number of samples were selected from each partition without replacement. The partitioning is to ensure that the data is relatively representative, and aims to prevent randomly choosing many nearby data points. Data was not replaced in the sampling to prevent selecting the same piece of data multiple times, further improving the representativeness of the sampling (3). Each partition has about 1078 samples, and 250 samples are selected from each partition, almost 24% of the partition. If data replacement was used there would be a non-negligible chance of reselecting the same data.

**Time (S) of Raw (Non-Sampled) Clustering Algorithm Prediction ( 21578 samples,default params)**

|  |  |  |
| --- | --- | --- |
|  | Euclidean | Cosine |
| K-Means | 30.2773 | 27.8734 |
| DBSCAN | 137.5356 | MemError |

**Time (S) of Sampled Clustering Algorithm Prediction (5000 samples, default params)**

|  |  |  |
| --- | --- | --- |
|  | Euclidean | Cosine |
| K-Means | 7.7462 | 8.5513 |
| DBSCAN | 11.7726 | 4.2628 |

The sampled data was then vectorized to allow clustering to occur. The word (feature) and class vectors of each document were vectorized before being fed to the SKLearn clustering toolkit. Word/feature vectorization was done by first finding all unique words in the processed/reduced article data, defining a consistent ordering of the values, and filling in the feature vector with the word's TF-IDF value if it was found in the document. Class labels were simply transformed into integers. In the original corpus, each document can have 0 or more class labels, but after vectorization, documents without labels were assigned the “0” label to allow for easier manipulation of data.

SKLearn's packages were used for the clustering implementations. The two clustering algorithms used were K-Means (Euclidean) and DBSCAN. K-Means is a fast, relatively simple clustering algorithm that separates a given number of clusters based upon distance from a calculated centroid. Initial centroid choices are important as poor centroids can lead to poorly clustered data (as the centroid can lie between data). SKLearn uses a default of 10 different centroid seed restarts which are chosen “in a smart way” using its k-means++ algorithm (4). K-means++ likely runs k-means on a smaller, representative data sample. DBSCAN is another clustering methodology that finds core data points of high density and expands clusters outwards from them. It is a strong algorithm for finding clusters of similar density, is resistant to noise, and can create irregularly shaped clusters (5). DBSCAN finds core points based upon it neighbors: there must be a minimum number of neighbor points within an epsilon distance. SKLearn uses NearestNeighbor algorithm modules to find neighboring points.

One drawback of SKLearn is the lack of Spherical K-Means, a K-Means algorithm with Cosine distance metric. In this lab, Spherical K-Means was emulated through usage of L2 normalization of data and SKLearn's Euclidean K-Means. Usually, spherical K-Means implies a cosine distance function and the constraint that centroid vectors are forced to lie on the unit circle (L2 normalized). SKLearn does not have a default Spherical K-Means implementation, but using L2 normalization on the original data, the default K-Means becomes proportional to the Spherical K-Means, and the results are similar (2). L2 normalizing the data before clustering makes Euclidean K-Means proportional to the Cosine Distance function. When the magnitude of direction/feature vectors are constrained to equal 1, the square of the Euclidean distance is equal to twice the magnitude of the cosine distance.

Proof of the Relationship Between Squared Euclidean Distance and Cosine Distance (2)

This initial normalization of the data is important as it allows the subsequent Euclidean K-Means to achieve results proportional to Spherical K-Means. As the results are proportional, clusterings will be similar (2). Spherical K-Means typically performs better on text classification as representative feature vectors are classically very sparse (3).

A variety of metrics were used for clustering quality. K-Means is commonly evaluated based upon the varying clusters' Sum of Squared Error values. SSE sums the squared euclidean distance of every point from its centroid's midpoint. Average cluster radius is a similar measure and was also calculated. K-Means tends to create globular clusters, so calculations measuring point distance to its centroid do a fair job of informing on the relative quality of the clustering. In DBSCAN however, clusters can be irregularly shaped and thus radius and SSE are not useful measures. Entropy was thus used as a cluster quality measure for both algorithms. Entropy measures the homogeneity/spread of different classes within a cluster. The Entropy of the original, unclustered data was found and used in conjunction with the Entropy of the clustered data to determine Information Gain of each clustering schema.

Different clustering schema were compared to find the optimal parametrization of the clustering algorithm. The original, un-sampled data has 121 class labels (including the No-Class label) while the random, representatively sampled version tended to fluctuate around 85-95 classes. If the classes are assumed to be internally similar, it would make sense if the number of clusters were close to the number of classes. However, as more clusters are added to K-Means the run-time of the algorithms increases, so there is a trade-off when using this algorithm. Using a saved representative sample, different clustering parameters were compared across several runs of the algorithms.

**K-Means Clustering Quality**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #Clusters | #classes\*2 (184) | #classes\*1.2 (110) | #classes (92) | #classes\*.8 (73) | #classes/2 (46) |
| K-Means | T= 39.833  IG= 0.6600  R= 696.0  SSE= 312974  SD= 33.7 | T= 24.899  IG= 0.5272  R=813.91  SSE=549325  SD= 51.1 | T= 23.128  IG= 0.4915  R= 905.69  SSE= 666271  SD= 41.35 | T= 19.468  IG= 0.4278  R= 892.72  SSE= 856168  SD= 67.3 | T= 16.811  IG= 0.3417  R= 961.61  SSE= 1406545  SD= 127.8 |
| K-Means (Cosine) | T= 40.803  IG= 0.7075  R= 103.65  SSE= 5508  SD= 29.9 | T= 23.371  IG= 0.6055  R= 123.90  SSE= 9554  SD= 17.2 | T= 24.021  IG= 0.5661  R= 125.23  SSE= 11593  SD= 24.0 | T= 18.586  IG= 0.4744  R= 126.48  SSE= 14913  SD= 26.9 | T= 13.493  IG= 0.3419  R= 127.65  SSE= 24653  SD= 49.8 |

Time (s), Information Gain, Average Cluster Radius, Average Cluster SSE and Cluster Size Standard Deviation

**DBSCAN Clustering Quality (Min\_Samples)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| min\_samples (eps=default) | 1 | 2 | 4 | Default |
| DBSCAN | T= 9.552  IG= 3.387  R= 5.71 \*10^-17  SSE= 2\*10^-31  SD= 0  #C= 4907 | T= 9.531  IG= 0.0629  R= 19.31  SSE= 1408987  SD= 642  #C= 56 | T= 9.525  IG= 0.0098  R= 179.54  SSE= 13350511  SD= 1845  #C= 6 | T= 9.518  IG= 0.0064  R= 269.19  SSE= 20040605  SD= 2146  #C= 4 |
| DBSCAN (Cosine) | T= 1.401  IG= 1.48  R= 43.095  SSE= 20894  SD= 60.1  #C= 2032 | T= 1.390  IG= 0.188  R= 517  SSE= 440509  SD= 251  #C= 170 | T= 1.388  IG= 0.05616  R= 736  SSE= 3137741  SD= 660  #C= 25 | T= 1.393  IG= 0.0390  R= 770.98  SSE= 5630207  SD= 858.6  #C= 14 |

Time (s), Information Gain, Average Cluster Radius, Average Cluster SSE, Size Standard Deviation, and Num Clusters

**DBSCAN Clustering Quality (Eps, min\_samples=2)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| eps | .54 | .53 | .5 | .45 |
| DBSCAN (Cosine) | T= 1.396  IG= 0.1255  R= 547  SSE= 804000  SD= 381  #C= 96 | T= 1.412  IG= 0.1401  R= 536  SSE= 665924  SD= 336  #C= 115 | T= 1.398  IG= 0.11  R= 570  SSE= 870205  SD= 408  #C= 170 | T= 1.404  IG= 0.2685  R= 489  SSE= 344138  SD= 218  #C= 212 |

Time (s), Information Gain, Average Cluster Radius, Average Cluster SSE, Size Standard Deviation, and Num Clusters

The K-Means table shows the clustering quality across all trial runs. For every run the approximate cosine K-Means results in greater information gain and smaller cluster size standard deviation. Number of clusters are assigned relative to the number of classes in the sampled data, which was 92 for the given data sampling. As the number of clusters increased, so did the time it takes for K-Means to run (as expected, time complexity is O(#clusters\*#samples\*#iterations)) as well as the information gain. Typically, more clusters equates to larger IG as it allows each cluster to be relatively more pure, but there reaches a point where the time trade-off is not worth it. For K-Means, it appears the ideal number of clusters is #classes \* 1.2. It has the most IG with the least amount of error and StdDev alongside marginal time increase over #classes. It is surprising that the number of clusters exceeds the number of classes, but it could just be due to over-fitting of the data. It could also be due to that fact that documents that have multiple class labels could also be more similar to documents with an equivalent list, rather than any individual label. This would cause the number of expected clusters to increase by the number of unique, multi-class document labels.

The DBSCAN table shows the clustering quality across all trial runs. The time it takes for DBSCAN to run is remarkably consistent across all runs. This is likely due to the implementation of SKLearn relying heavily on memory: it bulk computes all neighborhood queries, memory is O(#data \* avg #neighbors). Since the number of data points does not change, and it does not rely on the number of clusters, the time cost is very similar throughout. A series of trials were ran to find the “best” values of min\_samples and epsilon, which were found to be 2 and 0.53 respectively. These two values resulted in the most information gain with a number of clusters near the expected class number. Most values resulted in one cluster or one cluster per data value.

Overall, K-Means was found to be a better classifier than DBSCAN for this data. SSE and radius are not very applicable to DBSCAN due to its non-globular nature, and thus Information Gain was the main metric used to determine best classifier. In K-Means the information gain was much better than DBSCAN, when a similar number of clusters are created. K-Means also uses much less memory than DBSCAN, though is a fair amount slower (though still pretty fast),

Sources:

1. “Margin-based local regression for adaptive filtering”. Proceedings of the 12th International Conference on Information and Knowledge Management, pages 191-198, 2003. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.307.1537>

2. Ashok N. Srivastava, Mehran Sahami. “Text Mining: Classification, Clustering and Applications”, pages 162-163. [https://books.google.com/books?id=BnvYaYhMl-MC&pg=PA162&lpg=PA162&dq=clustering+l2+normalization+spherical+k+means&source=bl&ots=oj1IxpWWgg&sig=7r7u\_3gCwBQu9NgJg-nrZrCgPyo&hl=en&sa=X&ved=0CEEQ6AEwBGoVChMIzerw8aH6yAIVyz8-Ch3UDg2c#v=onepage&q=clustering%20l2%20normalization%20spherical%20k%20means&f=false](https://books.google.com/books?id=BnvYaYhMl-MC&pg=PA162&lpg=PA162&dq=clustering+l2+normalization+spherical+k+means&source=bl&ots=oj1IxpWWgg&sig=7r7u_3gCwBQu9NgJg-nrZrCgPyo&hl=en&sa=X&ved=0CEEQ6AEwBGoVChMIzerw8aH6yAIVyz8-Ch3UDg2c" \l "v=onepage&q=clustering l2 normalization spherical k means&f=false)

3. Mary Parker, “Sampling With Replacement and Sampling without Replacement”. <https://www.ma.utexas.edu/users/parker/sampling/repl.htm>

4. “sklearn.cluster.Kmeans”. Scikit Learn. <http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

5. “sklearn.cluster.DBSCAN”. Scikit Learn. <http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html>