CSE 5243: Introduction to Data Mining Assignment 5

Vaibhav Devekar (devekar.1)  
Akshay Nikam (nikam.5)

# Goal of the Assignment

This assignment aims at demonstrating classification based on association rules as well as using clustering for improving the performance of the classification

# Performance Metrics

1. Accuracy by at least one matching

This is an optimistic metric. It is based on having at least on match between predicted labels and actual labels for each test instance. We then average over the entire test instance.

1. Accuracy by one-to-one matching

In this case, we count the number of same pairs between predicted and actual and divide it by the least of sizes of predicted and actual labels. We then average it over all test instances.

1. F-measure(Harmonic mean between Precision and Recall)

Where,   
a = True Positive, b = False Positive, c = False Negative

# Assumptions

1. K = 5, since the number of class labels for each article is 3- 5.  
   Additionally, we also apply rules until we get the number of labels equal to the actual number.
2. The clusters from K-means algorithm were obtained using Cosine distance metric. The Cosine metric was found to give the best performance in previous assignment.
3. We also identify the cluster for test instances in Algorithm 2 using Cosine metric.
4. Only topics are considered as class labels. The dataset, thus, consists of 11305 records on which we perform 80-20 split.

# ALGORITHM 1

The algorithm was implemented with two stages:

1. Train Phase

After reading in the transaction matrix, we perform the following steps:

1. Write the matrix to file to pass as input to the Apriori program.
2. Invoke the Apriori program
3. Read the rules from the output file, discarding ones which do not match the form W -> C, where W are the words while C is a class label.
4. Order the rules and subsume it.
5. Collect the transactions that were not covered by the rules and repeat the above steps for these transactions until most are covered.
6. Test Phase

Using the subsumed rules from above phase, we apply it on the test set. As stated before, we apply the rules twice; one with a fixed K and in other case, we keep applying rules until we get labels equal to the actual number. If no rule is applicable, we apply the default label from the transactions that were not covered in train phase.

## Results and Performance:

For results and performance computation, we considered topics and place labels of each article in each cluster and used following metrics to evaluate performance and the quality of clustering.

### Entropy:

For finding out how the algorithm performed, we used entropy as the performance metric. It is a measure of uncertainty and depicts the number of bits required to represent a point in the given data.

Computing entropy:

Entropy is first individually computed for each cluster and then a weighted average is taken based on the number of points in the clusters. Since each article has multiple topics and places, we gave each label of the article an equal weight, such that the total weight of all topics and place labels for an article is always 1. For example, if an article has 2 topics and 1 place, each are given 1/3 weight. Now, each topic’s/place’s weight is summed over all the records in that cluster. Once we get a vector of total weights of all class labels, we normalize it to find probabilities of each topic/place. Then entropy for the cluster is computed as:

   \displaystyle
   H(X)= - \sum_{i=1}^np(x_i)\log_b p(x_i)

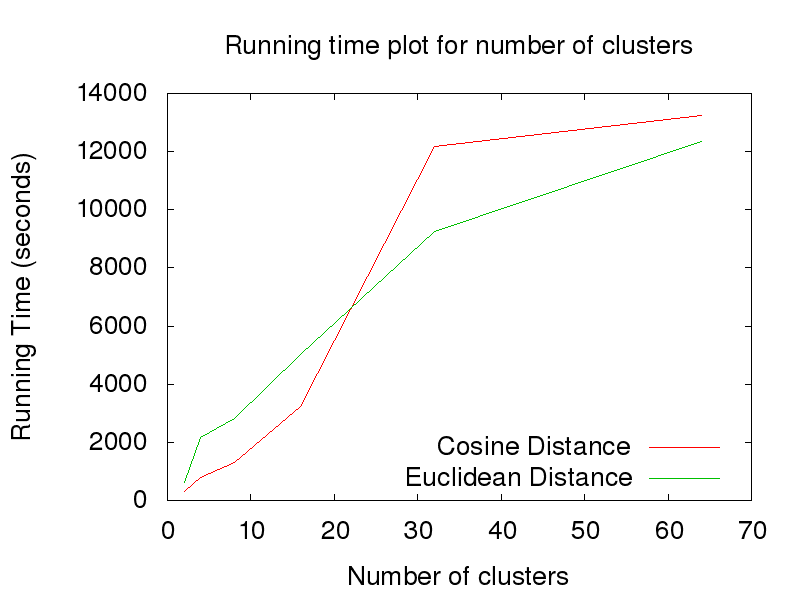
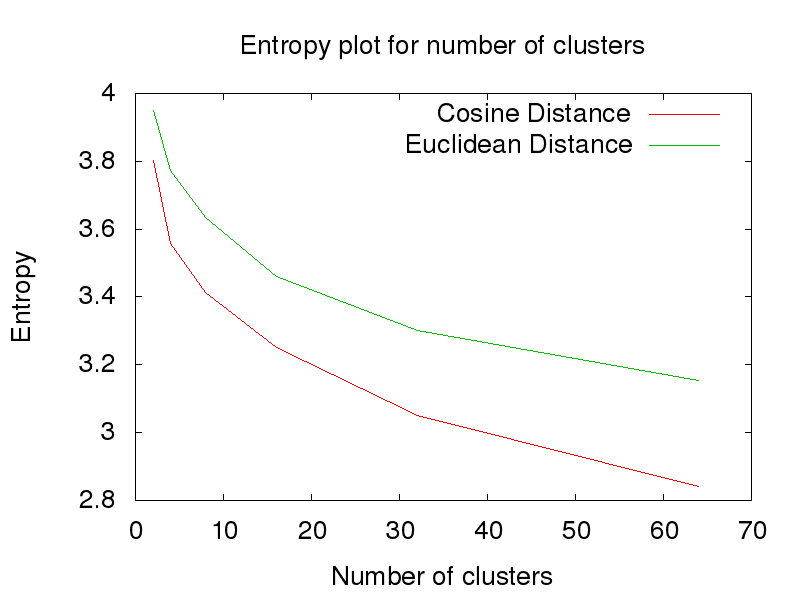

For n class values and b=2.

We found that as the number of clusters increase, entropy gradually decreases, meaning a better quality of separation of articles.

For K-means clustering algorithm, we found following entropies:

|  |  |  |
| --- | --- | --- |
| **# Clusters** | **Entropy for Cosine Distance** | **Entropy for Euclidean Distance** |
| 2 | 3.80375332771 | 3.94993482586 |
| 4 | 3.55697535925 | 3.77195219452 |
| 8 | 3.41406922379 | 3.63348425172 |
| 16 | 3.25008232272 | 3.46016354507 |
| 32 | 3.05113063415 | 3.29978197132 |
| 64 | 2.84177028704 | 3.23467349883 |

According to the following plot, it can be observed that Cosine distance produces lower entropy and hence is a better distance metric for clustering given data.



### Scalability:

The running time is roughly directly proportional to the number of clusters K provided as input. We took results for K=2, 4, 8, 16, 32, 64. It is found that running time is directly proportional to the number of clusters and the number of iterations required to converge. The above plot shows the results for the same.

# ALGORITHM 2

We reused the code Algorithm 1, albeit with a few modifications. We also use the Kmeans code from Assignment 4 to obtain the clusters. The algorithm consists of following steps:

1. Clustering

We use K means code to obtain the clusters for various number of clusters with Cosine distance. The output is used to reconstruct transaction matrix for each cluster.

1. Train Phase

This phase is same as that for algorithm 1. In this phase, we apply the Algorithm 1 for transaction matrix of each cluster and store the subsumed rules.

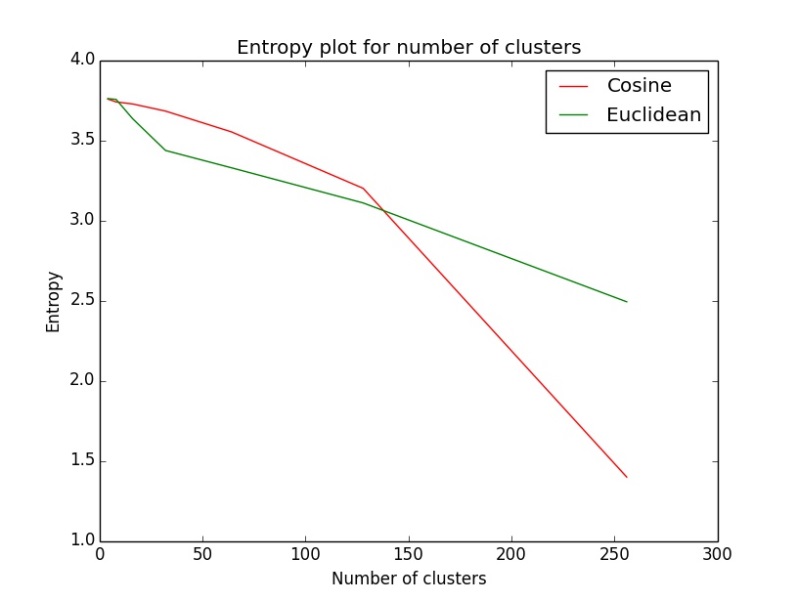
1. Test Phase

We derive the representative means for each cluster. Using Cosine metric, we identify the cluster to which each test instance belongs. We apply the subsumed rules from that cluster on the test instance to obtain the class labels.

### Entropy:

The clusters are skewed with most articles (70%) in a few clusters and the remaining distributed among the other clusters. A sample result and comparison is provided in ‘CLUTO’ section below.

|  |  |  |
| --- | --- | --- |
| **# Clusters** | **Entropy for Cosine Distance** | **Entropy for Euclidean Distance** |
| 4 | 3.75693066406 | 3.75960564751 |
| 8 | 3.73908154017 | 3.75483842574 |
| 16 | 3.7263998765 | 3.6347700202 |
| 32 | 3.6818774862 | 3.4361708798 |
| 64 | 3.55229254978 | 3.32822322466 |
| 128 | 3.19986385296 | 3.10870895911 |
| 256 | 1.39849314748 | 2.49199695245 |



We observe that the cosine metric produces lower entropy when the number of clusters is high.

Scalability:

|  |  |
| --- | --- |
| Stage | Time in seconds |
| 1 | 26.0988128689 |
| 2 – Cosine | 475 |
| 2 – Euclidean | 754.48 |
| 3 | 2747.63 |
| 4 | 29.0881068614 |

Since stage 3 works directly on distances, the choice of metric does not affect its running time. Computation time for cosine distances is less than that of Euclidean since it uses magnitudes which can be pre-computed and reused. Computation time is considerably reduced since only 11305 features are used instead of entire dataset.

# Other Approaches

Initially, we also included Places in the class labels.

# Individual Contributions

Vaibhav implemented the basic algorithm for both algorithms. Akshay implemented performance metrics as well as analyzed the performance for both algorithms.

REFERENCES: http://www.borgelt.net//apriori.html