HW3: Text Clustering

Bhavika Tekwani
btekwani@gmu.edu

Introduction

Implementing K-Means clustering is challenging for two reasons. One being, the selection of 'k' and the initial centroids and the second being, the evaluation of results considering unsupervised learning does not offer simple measures to evaluate the *goodness* of our results the way supervised learning does (accuracy, F1 score, etc). I implemented K-Means clustering in Python. I used scikit-learn for feature extraction and dimensionality reduction. On the Iris dataset, I used measures like homogeneity, completeness and silhouette coefficient to analyse results.

Rank and Accuracy

As of 10/22/2016 at 10:58 PM, my rank was 3 and V-score was 0.55 on the Text Clustering leaderboard. On the Iris Clustering leaderboard, I held rank 10 and my V-score was 0.73.

Approach

Feature selection and dimensionality reduction

Borrowing from HW1, I used TF-IDF to vectorize the text data at hand. The dimensions of the original featureset are 8580 * 57940. I've used Principal Component Analysis (PCA) to reduce the dimensionality of our featureset. Setting the number of components to 10 in PCA showed the best explained variance. The explained variance is an indicator of how many components are needed to explain most of the variance in the data. Using PCA to deal with the curse of dimensionality helped me obtain a reduced featureset of size 8580 * 10.

K-Means

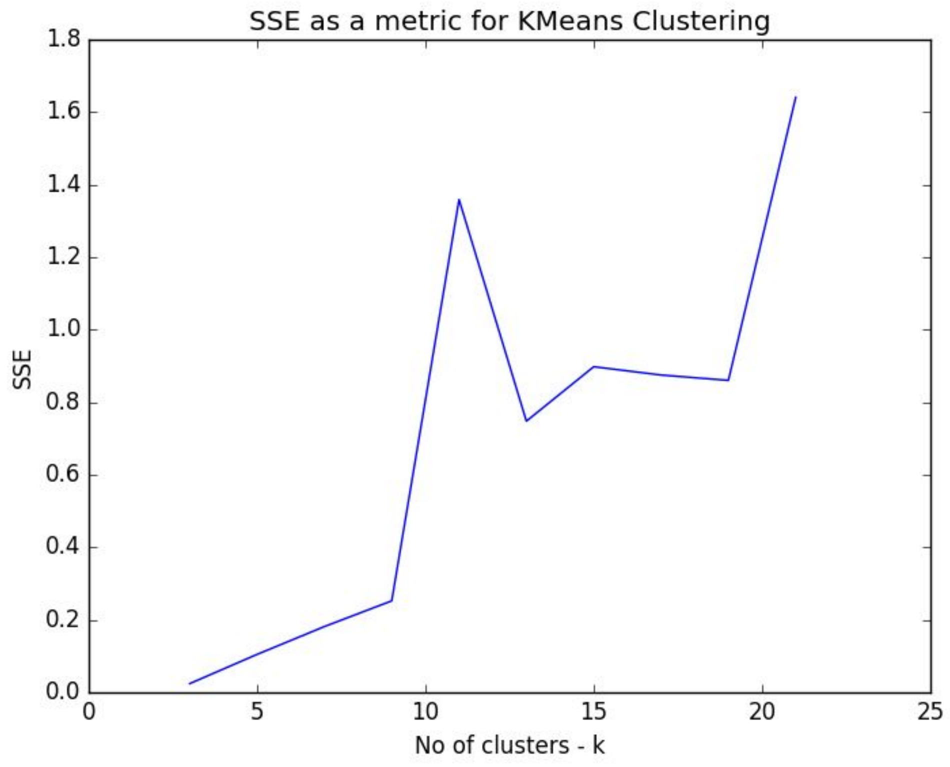
I've implemented Lloyd's k-means algorithm and used it for both Iris and text datasets. The only difference is the number of clusters for each dataset. I tried two distance measures - cosine and Euclidean. Euclidean distance lends itself better to both datasets.

Pseudocode:

X : set of points to be clustered
k : number of clusters
n: number of iterations
C: the cluster centroids
l : length of X
P : {p(i) | i = 1, 2, ….l }, the labels for X

```
kmeans(X, k, n)
    $C_{i-1} \leftarrow C_i$                              $C_i$ is selected randomly
    for all i $\varepsilon$ (1, n) do
        p(i) $\leftarrow$ argmin d($x_i$, $C_j$)                 j $\varepsilon$ [1...k]
    for all j $\varepsilon$ (1, k) do
        $C_j \leftarrow$ mean($X_i$), whose p(i) = j
    repeat until $C = C_{previous}$
```

Internal Evaluation Metric: On the Iris dataset, I used measures like homogeneity, completeness and V-score to evaluate my K-Means implementation. I achieved a V-score of 0.73 on the leaderboard and 0.74 when I evaluated my results independently. On the text dataset, I used Sum of Squares Error (SSE) to evaluate the performance of KMeans. The graph below shows the trend of SSE versus k where k goes from 3 to 21 in steps of 2. Ideally, a graph of SSE versus k is in the shape of an elbow where the bend signifies the best value of k.



References

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. "Scoring, Term Weighting
and the Vector Space Model."Introduction to Information Retrieval. New York: Cambridge UP, 2008. N. pag. Web.

Rose, Brandon. "Document Clustering with Python." Document Clustering with Python. N.p., n.d. Web. 23 Oct. 2016. <http://brandonrose.org/clustering>.

Piech, Chris. "K Means." CS221. N.p., n.d. Web. 23 Oct. 2016. <http://stanford.edu/~cpiech/cs221/handouts/kmeans.html>.

Gove, Robert. "Using the Elbow Method to Determine the Optimal Number of Clusters for K-means Clustering." Popular Blocks. N.p., n.d. Web. 23 Oct. 2016. <https://bl.ocks.org/rpgove/0060ff3b656618e9136b>.