EN.520.666 Information Extraction from Speech
Homework 1

February 6, 2025

Prabhav Singh
psingh54@jhu.edu
MSE CS

1. **Readings**

Ans: I have read Chapter 1 from the 'Statistical Methods for Speech Recognition' book! I have also read the paper recommended!

2. **Computer Exercise**

Part (a)

PLOTS

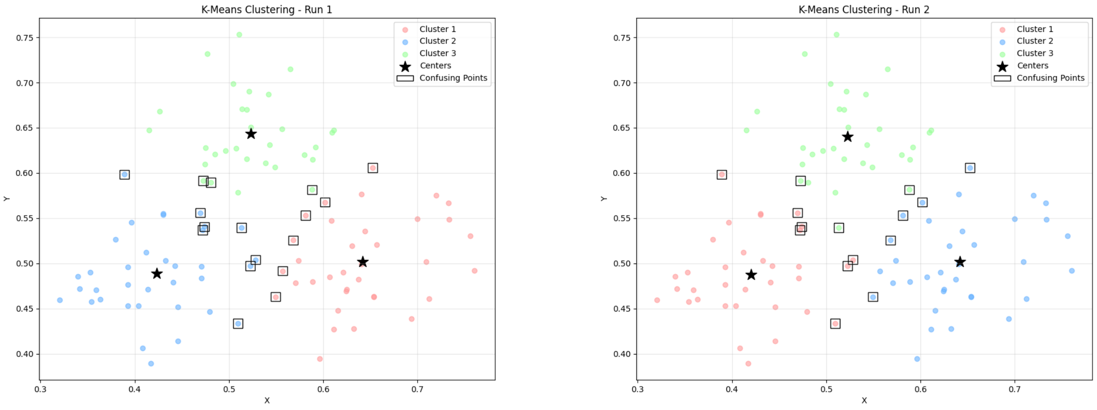

Figure 1: Cluster assignments from two different runs of K-Means.

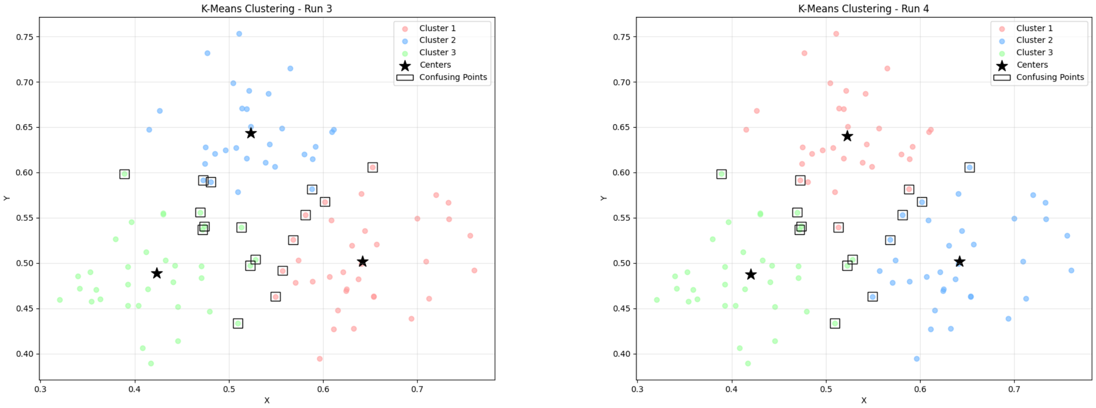

Figure 2: Cluster assignments from two additional runs of K-Means.

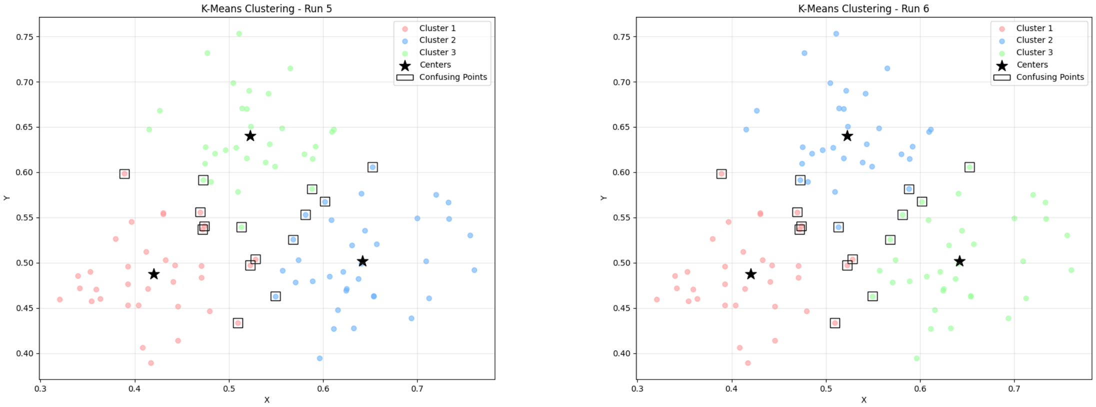

Figure 3: Cluster assignments from two additional runs of K-Means.

Ans: For the code, please refer to the GitHub repository linked here (`https://github.com/Prabhav55221/520.666-S25`). Answers and discussions for each question are given below. For reference, the centers found were very close in all runs. These did not differ much except by orders of 0.01. This is understandable since the points do follow natural clusters. These cluster centers are marked with the star!

Observations & Discussions

- **Common Tendencies:** The clustering algorithm generally converges to similar groupings of points, with clusters forming around dense regions in the dataset. Despite different initializations, the final clusters appear consistent in structure. The centroids move iteratively to optimize Euclidean distances, and most runs yield nearly identical groupings.

- **Outlier Behavior:** Occasionally, a point that is equidistant from two centroids may switch its cluster assignment depending on the initialization. These points are marked with the squares. **We can take the example of the square bang in the center** - That point switches cluster assignment almost every time randomly. From the six runs, the point is assigned basically randomly to any one of the cluster. This is because it is almost equidistant from the centers!
- **Problems with K-Means:** Some issues with KMeans are:
  1. One of the biggest challenges observed is sensitivity to the initial choice of cluster centers. Different initializations led to minor variations in the final clustering, though the main clusters remained similar. This highlights K-Means reliance on good initialization, and using techniques such as K-Means++ may help improve stability.
  2. K-Means assumes spherical clusters, which may not be ideal for datasets with non-convex distributions.
  3. Another drawback of the K-means algorithm is that we have to set the number of clusters (K) in advance. Choosing an incorrect number of clusters can lead to inaccurate results.
  4. The time complexity of the algorithm depends on the number of clusters and also the numbers of dimensions in data. So, even moderately large datasets can be challenging to handle if they're high-dimensional.

3. **Product Quantization**

Part (a)

Ans: Space Complexity

Assume we K Centers, D dimensional data and N data points.

For K-Means, the space complexity is $\mathbf{O}(KD)$. Actually it would be $\mathbf{O}(KD + ND)$ because we need to store the points for each center also but the ND can be ignored since it is common to both methods.

However, for Product Quantization, we spit the the original dimensionality of the data into $D' = D/m$, where m is the number of subspaces we create. What this does is allow us to choose a $k << K$ for each subset. This makes the space complexity: $\mathbf{O}(m.k.\frac{D}{m}) = \mathbf{O}(kD)$.

Further, the paper on PQ (*Jégou et al., Product Quantization for Nearest Neighbor Search. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011*) says that this k is actually orders of magnitude smaller than K - $k = K^{1/m}$.

Hence:

$$KMeans : \mathbf{O}(KD)$$

$$PQ : \mathbf{O}(K^{1/m}D)$$

We can see that we reduce the space complexity by orders of m magnitude - Where m is selected by us!

Time Complexity

Using the same principle, the time complexities are (Technically it should $O(NKD)$ and $O(NK^{1/m}D)$, but the N is common so we can ignore it.):

$$KMeans : \mathbf{O}(KD)$$

$$KMeans : \mathbf{O}(K^{1/m}D)$$

Usefulness

Product quantization dramatically reduces both the memory and computational costs compared to standard K-Means. It has below benefits:

1. Because product quantization allows for very large effective codebooks (via the Cartesian product of small sub-codebooks) without incurring a proportional increase in computational cost, we can achieve a good balance between the quantization error (accuracy) and computational efficiency.
2. The paper ‘Product Quantization for Nearest Neighbor Search’ demonstrates that PQ can significantly accelerate search in high-dimensional spaces while keeping the memory usage low.
3. The efficiency makes it feasible to apply quantization-based approaches to very large datasets - specifically mage retrieval systems (as mentioned in the paper).