ML- HW4 Theory

Amirhossein Izadi | 99243018

Q1.

The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high dimensional spaces that do not occur in low-dimensional settings. In other words, problems with high dimensionality result from the fact that a fixed number of data points become increasingly "sparse" as the dimensionality increase.

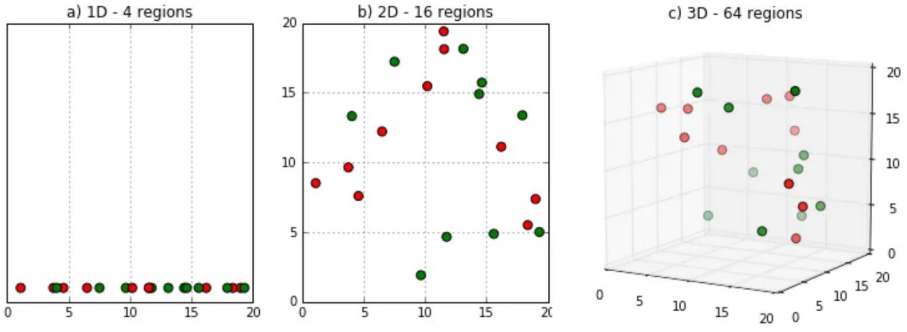

Figure 1.1: image from heartbeat.comet.ml

For clustering purposes, the most relevant aspect of the curse of dimensionality concerns the effect of increasing dimensionality on distance or similarity. In particular, most clustering techniques depend critically on the measure of distance or similarity, and require that the objects within clusters are, in general, closer to each other than to objects in other clusters. Otherwise, clustering algorithms may produce clusters that are not meaningful.

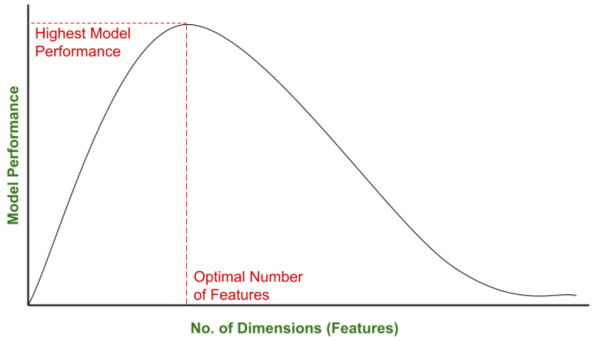

Figure 1.2: image from GeeksforGeeks

In principle, there are two methods for counteracting the curse of dimensionality: feature selection and feature engineering (feature extraction). Feature selection encompasses techniques such as Filter Selection Methods, Wrapper Feature Methods, and Embedded methods. On the other hand, feature engineering involves computationally altering features to reduce their dimensionality, utilizing techniques such as PCA, SVD, and others.

This answer is written by considering the following sources (some parts are copy directly). You can consider each one for further details : source1 - source2 - source3

Q2.

Determining the optimal number of clusters is a big step in some clustering. we can use some methods include direct methods and statistical testing methods.

Direct methods

Consists of optimizing a criterion, such as the within cluster sums of squares or the average silhouette. The corresponding methods are named elbow and silhouette methods, respectively.

Statistical testing methods

consists of comparing evidence against null hypothesis. An example is the gap statistic.

Elbow method

---

elbow method gives us an idea on what a good k number of clusters would be based on the sum of squared distance (SSE) between data points and their assigned clusters' centroids. We pick k at the spot where SSE starts to flatten out and forming an elbow.

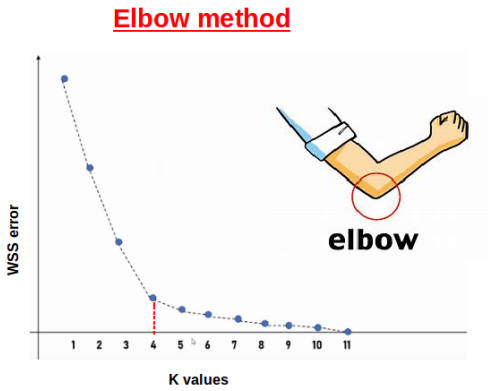

Figure 2.1: image from meduim.com

Unfortunately, we do not always have such clearly clustered data. This means that the elbow may not be clear and sharp.

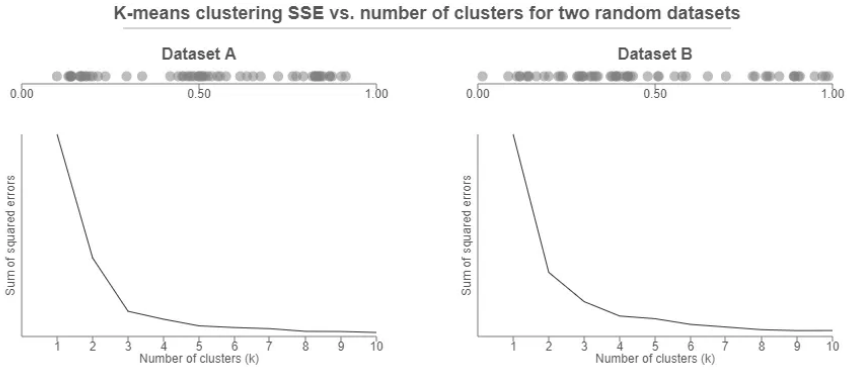

Figure 2.2: image from meduim.com

in figure2.2 for dataset A, the elbow is clear at k = 3. However, this choice is ambiguous for dataset B. We could choose k to be either 3 or 4. In such an ambiguous case, we may use the Silhouette Method.

This answer is written by considering the following sources (some parts are copy directly). You can consider each one for further details : source1 - source2

Q3.

In the context of compressing an image with three color channels (green, red, and blue), k represents the number of clusters or colors we want to use to represent the image.

Data Preparation: Flatten the image matrix into a list of pixel values, where each pixel is represented by a triplet (R, G, B), indicating the intensity of red, green, and blue channels.

```
rows = image.shape[0]
cols = image.shape[1]

image = image.reshape(rows*cols, 3)
```

Apply k-means: Apply the k-means algorithm to cluster these pixels into k clusters. The algorithm will assign each pixel to the cluster whose centroid is closest to it.The centroids represent the average color values of the pixels in each cluster.

```
kmeans = KMeans(n_clusters= 5) #here we use just 5 clusters (color category )
kmeans.fit(image)
```

Quantization: Replace the original pixel values with the centroid values of the clusters to which they were assigned. This is the quantization step.Instead of storing each pixel's RGB values, you store the k centroids and an index for each pixel indicating which centroid it belongs to.

```
ompressed_image = kmeans.cluster_centers_[kmeans.labels_]
compressed_image = np.clip(compressed_image.astype('uint8'), 0, 255)
```

Compression: Encode the indices and the centroid values, which will be smaller in size compared to the original pixel values. The image is now compressed with a reduced color palette.

```
compressed_image = compressed_image.reshape(rows, cols, 3)
```

Figure 3.1: image from cse.buffalo.edu

This answer is written by considering the following source and also some other parts might be AI generated. You can consider each one for further details : source1 - source2

Q4.

Due to the data shape and its noise we must use density based clustering models among conectivity based and centroid models. K-Means(centriod model) and Hierarchical(connectivity model) Clustering both fail in creating clusters of arbitrary shapes. They are not able to form clusters based on varying densities. That's why we need DBSCAN clustering.(density model) DBSCAN works on the idea that clusters are dense groups of points. These dense groups of data points are separated by low-density regions. The real-world data contains outliers and noise. It can also have arbitrary shapes (as shown in figures below), due to which the commonly used clustering algorithms (like k-means) fail to perform properly.

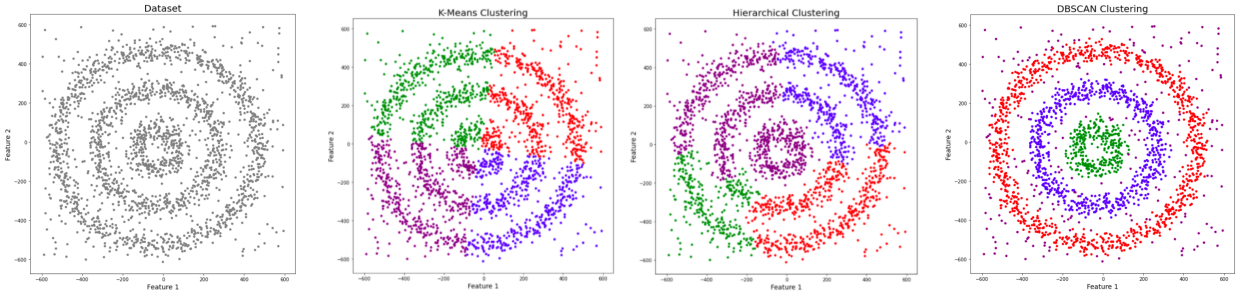

Figure 4.1: image from analyticsvidhya.com

This answer is written by considering the following source. You can consider for further details: source1