Clustering Chapter

I. Opening story

II. Clustering as a concept

Perhaps the most powerful pattern recognition machine is the human brain, using the signal collected by the eye. Without knowing context, this potent duo can pick out patterns that form the basis of knowledge and inference. And on their own, they are remarkable instruments to help us humans navigate our world. But, as we all experience, it is not easy to clone oneself to do human tasks at scale.

There is an abundance sociodemographic data published by statistical agencies around the world, but for the data to be more locally applicable and actionable, it needs to be more geographically granular. There are restrictions placed on governments publishing data in order to protect the privacy of individuals, thus resulting in coarse units of reporting. There are, for example, plenty of uses of where are people located within a zip code or county. Urban planners could better zone. Emergency responders can have a better sense of building assets and density of people in order to better plan respond to a natural disaster.^[Risk analysis paper citation]

Using nighttime satellite imagery from the Suomi NPP satellite, one could simply trace the outline of each city using our eyes as a guide, then apply the outline as a mask to refine the geographic shape of publicly available data. But manually doing this task is simply not scalable to billions of pixels of imagery. Like all things data science, we could apply clustering techniques at scale, extracting the areas in satellite imagery that emit a distinguishable level of light.



This raises the question: what defines the border between light and dark? This is a strikingly similar idea to separability, a measure of dissimilarity. If we think of each pixel as an element in a vector of light intensity, we could treat satellite imagery as data. This data in turn has statistical properties that can be exploited to scale our problem. Let’s take the case of San Francisco, which is a mass of human activity on a peninsula on the Pacific Coast. When we plot a histogram of logarithm transformed radiances from the satellite imagery, we find a bimodal distribution, which implies that there are at least two types of activity that reflected in this data – a mixture of distributions. A simple way to separate one distribution from the other is to define each activity by its mean, located at the peaks.

[2x2 plot – image of San Francisco, Histogram of San Francisco, Partitioned histogram, cut-out of San Francisco]

To classify each pixel as either light or dark, we can use a simple dissimilarity measure. For continuous values, dissimilarity can be as Euclidean Distance:

$$\d(x,y) = \sqrt{\sum\_i(x\_i, y)^2}$$

And calculate distance between each point and each peak, assigning each pixel to the closest peak. While this simplistic example is effective, it can be automated and scaled using one of the many commonly used clustering algorithms.

In this chapter, we explore two of the most commonly employed clustering algorithms: K-means clustering and hierarchical clustering. Each clustering technique is computationally intensive, constructed on different assumptions that help it accomplish this task, but the core differences between the techniques enable very different use cases.

# K-means

K-means clustering identifies observations that belong to a latent group by treating variable sets as coordinates in n-dimensional space. As the true value of \*k\* is not known, the user defines the parameter that controls the number of clusters that will be returned upon running the algorithm.

Given \*k\*, the goal is to assign each observation to a cluster \*C\*. Each cluster \*C\* is defined by a set of centroids that are the means of input variables X for each cluster set. The optimal set of clusters minimizes the total cluster variance:

$$min \sum\_{k=1}^{k} N\_k \sum\_{C(i)=k} |x\_i-\mu|^2$$

The steps of the algorithm are iterative and computationally intensive, but straight forward:

1. Randomly assign k-centroids;

2. Assign each point \*i\* to the closest cluster \*C\*;

3. Recalculate the centroid coordinates for each C;

4. Repeat steps 2 and 3 until point assignments no longer change

The first step involves selecting $k$-number of random centroids from the feature space and assigning each centroid a label. For each observation in the data, calculate the Euclidean distance from each point to each centroid, then assigning observations to the closest centroid. This is known as the \*assignment\* step -- all points take the label of its closest centroid. It is unlikely that this initial assignment is likely suboptimal, thus the algorithm will \*update\* the centroid coordinates by calculating the mean value of each feature within each cluster. Upon doing so, this assignment-update procedure is iteratively repeated until the centroid coordinates no longer change between iterations.

We can see that the algorithm generates clusters, but does not produce coefficients, weights or classification rules. \*How does one apply the results to new data?\* The simple solution is to treat cluster centroids as a training set, then apply 1-Nearest Neighbor algorithm to map new observations.