EECS 16ML Introduction to Machine Learning Skills
Fall 2020 Note 25

K-Means Clustering

K-Means is a commonly used unsupervised clustering algorithm. You may have seen K-Means frequently used for a variety of applications such as:

- Segmenting customers to create individual customer group profiles for advanced marketing techniques.
- Image segmentation and compression techniques for aiding Computer Vision.
- Clustering types of security threats in Information Technology.

1 Introduction to K-Means Clustering

Let's develop some intuition for how K-Means works and what information it can provide before diving into the math and algorithmic structure. We will provide a brief motivation for the technique, aspects of supervision in the algorithm, and how we evaluate its performance.

1.1 Motivation

Before actually a dataset to any predictive or fitting framework, it is important to get an idea of the kind of data present as well as how distinguishable different regions of the dataset are from other regions, especially if we desire to apply our final algorithm to a classification task. This distinguishability may be immediately available in the dataset. For instance, if we have a dataset of electricity usage used for heating as a function of average winter temperature in the state and each datapoint represents a government building in either a northern or a southern state, then the data will likely be easily separable into two or three clusters: buildings in northern states that require more heating, buildings in southern states which require little to no heating, and perhaps a third cluster for buildings in geographically intermediate states. We may already have an idea of the category each of the 50 states might fall into based on the heat map below.

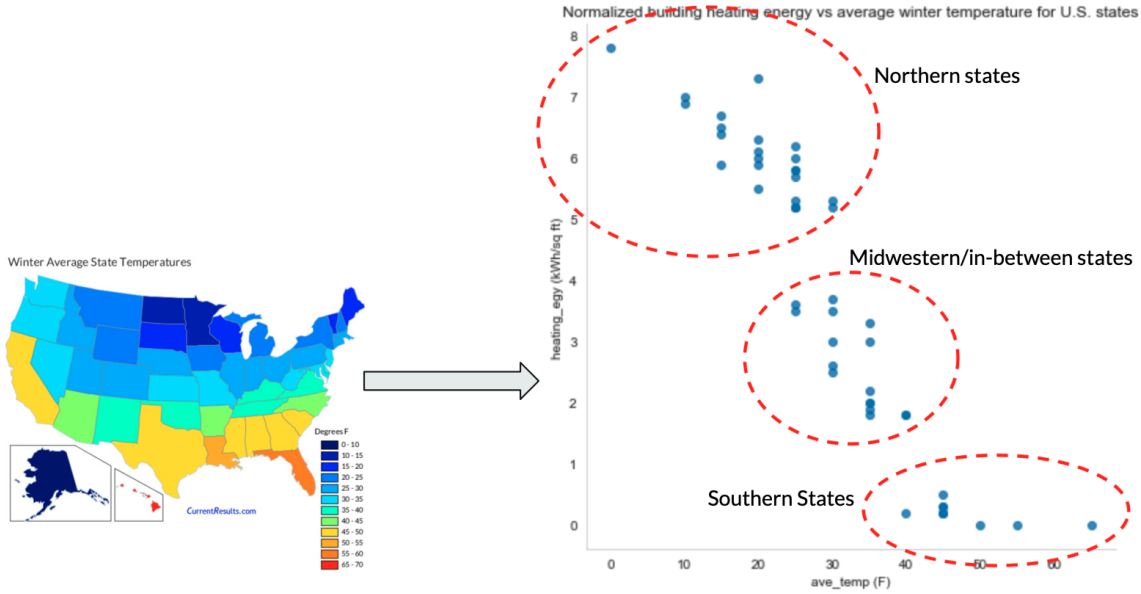

Figure 1: U.S. Heatmap and Clustering

In other cases, the delineations between datapoints may not be as clear, particularly if each data point is rich with features. For these, we can use dimensionality reduction to illuminate the differences between datapoints based on their principal components (taking advantage of the features with higher variance). We can use Principal Component Analysis (or PCA) for this dimensionality reduction task. If the data are easily distinguishable based on a plurality of their features, then PCA should readily separate the data points where nearby datapoints are similar according to their features.

Regardless of how the data is clustered, the heart of K-Means lies in the assumptions we make about the distribution of datapoints (e.g. circularly or spherically distributed, depending on the dimension of the data). With those in mind, we infer (systematically) where the "mean" for that particular cluster lies based on the distribution of points in that cluster. Below, you will see that we assign these means iteratively until the metrics we use to evaluate the clustering effectiveness no longer change.

1.2 Unsupervised Clustering

In most real-world categorization or clustering applications, the data provided to you will not be labeled with a category. Otherwise, we wouldn't need to apply an algorithm like K-Means to separate the categories based on their features! This means that K-Means is an *unsupervised* clustering method. We cannot guide its choice of cluster mean location except for in tuning the number of desired clusters and the dimensionality of the data considered (if dimensionality reduction is required). Additionally, this means that getting decent results from a K-Means implementation may require the input of a domain expert who could point out discrepancies in the way that the data has been clustered based on their experience in the field.

1.3 Choice of Cost

Because we do not have access to the true labels in the dataset, we need to come up with a way to inspect the performance of the algorithm as it iterates to convergence. It is most convenient to consider the L^2

distance between the datapoints. Using this distance, there are two ways to gauge how well the clusters are formed with K-Means: intercluster and intracluster similarity. We want low intercluster similarity (such that different clusters indeed represent different categories of datapoints) and high intracluster similarity (such that the datapoints within a cluster are more similar to each other than to points in other clusters). An intuitive way to think about this cost is as an energy expenditure of the individual datapoints, as if they were people belonging to a particular part of town. If we sent cars to pick up individuals from disparate parts of town based on clusters of homes, we would want to minimize the number of cars we would need to send out to people, so finding the optimal number of distinct clusters which have a noticeable separation is important (thinking in terms of *inter*-cluster distance). We would also want folks from one cluster to walk about the same distance to the pickup location (i.e. the mean of that cluster) as well as to minimize the distance people would need to walk from their homes to the car as the car made a stop at one cluster (thinking of *intra*-cluster distance.

2 How K-Means Works

The goal of K-Means clustering is two-fold:

1. Maximize the distance between clusters: we want each cluster to be as distinct; therefore, we want them to be as far away from each other as possible. We call this the inter-cluster distance.
2. Minimize the size of each cluster: we want all the data points assigned to each cluster to be as similar as possible; therefore, we want the distance between all the points in a single cluster to be as close to each other as possible. Typically, we optimize this by minimizing the distance of each data point to the center of the cluster. We call this the intra-cluster distance.

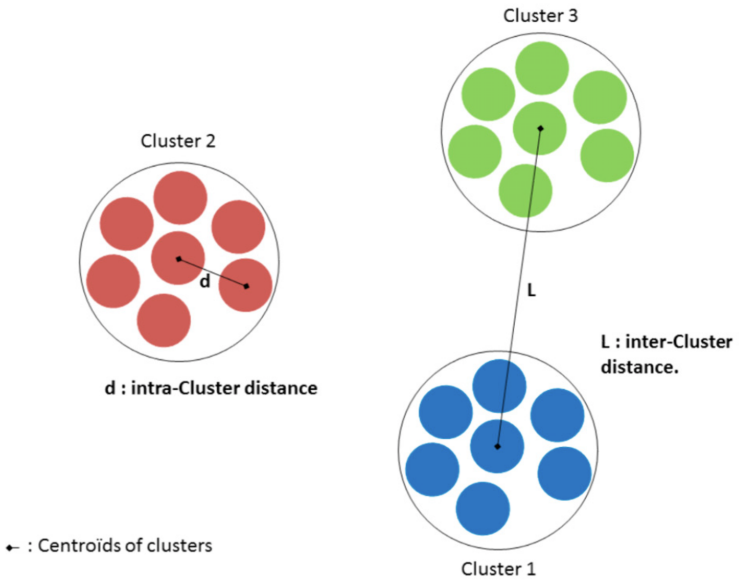

Figure 2: Intra-cluster vs Inter-cluster Distance [1]

[1] https://www.mdpi.com/2411-5134/4/1/17/htm

Formally, let X denote the set of N data points $\vec{x}_i \in \mathbb{R}^d$. A cluster assignment is a partition $C_1, ..., C_k \in X$ such that the sets C_k are disjoint and $X = C_1 \cup ... \cup C_k$. A data point $\vec{x}_i \in X$ is said to belong to cluster k if it is in C_k. K-Means solves the following optimization problem:

$$\underset{\{C_k\}_{k=1}^K, \{\vec{c}_k\}_{k=1}^K : X = C_1 \cup ... \cup C_K}{\arg\min} \sum_{k=1}^K \sum_{\vec{x} \in C_k} \|\vec{x} - \vec{c}_k\|^2$$

Unpacking the optimization problem, we observe that we are trying to find the cluster assignment that minimizes the sum of intra-cluster distances.

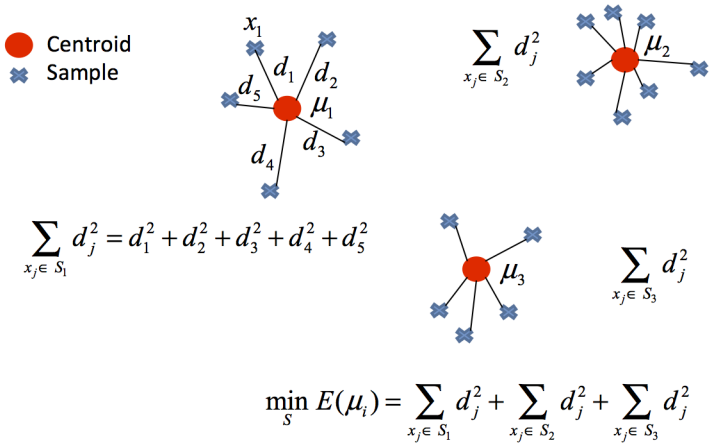

Figure 3: K-Means Objective Cost Function [2]

It is computationally quite difficult to try to optimize both inter and intra-cluster distances at the same time. Instead, we take a simplified approach where we repeatedly solve them one at a time and iterate until we converge upon an optimal solution. Imagine it this way: if we knew the cluster centers, then we can easily find the best assignment for each data point by assigning each data point to the closest cluster center. Conversely, if we knew the assignments for each data point, then we can easily find the cluster centers by finding the center of all the data points in that cluster (averaging all the data points).

This yields the algorithm we will use to find the centroids and cluster assignments:

Algorithm 1: Lloyd's Algorithm for K-Means

Initialize c_k, k = 1, ..., K;
while *intra-cluster distance keeps changing* **do**
  Given all c_k, update $C_1, ..., C_K$ by assigning each data point $\vec{x}$ to the closest c_k;
  Given the new $C_1, ..., C_K$, update c_k to be the average of all the data points in C_K;
end

At every step in the algorithm, the intracluster distance is guaranteed to either (1) decrease or (2) stay the same. At each step, we follow the 2-step process of finding cluster assignments, then finding the new centroids.

[2] https://www.unioviedo.es/compnum/labs/new/kmeans.html

A cluster assignment will only change between iterations if at least a single data point is now closer to a different centroid than in the previous iteration. This means that the distance contributed by that specific data point to the overall objective is going to decrease. Therefore, if we repeat the algorithm long enough, then we will eventually converge to a K-Means clustering assignment where our intra-cluster distance will stop changing between iterations.

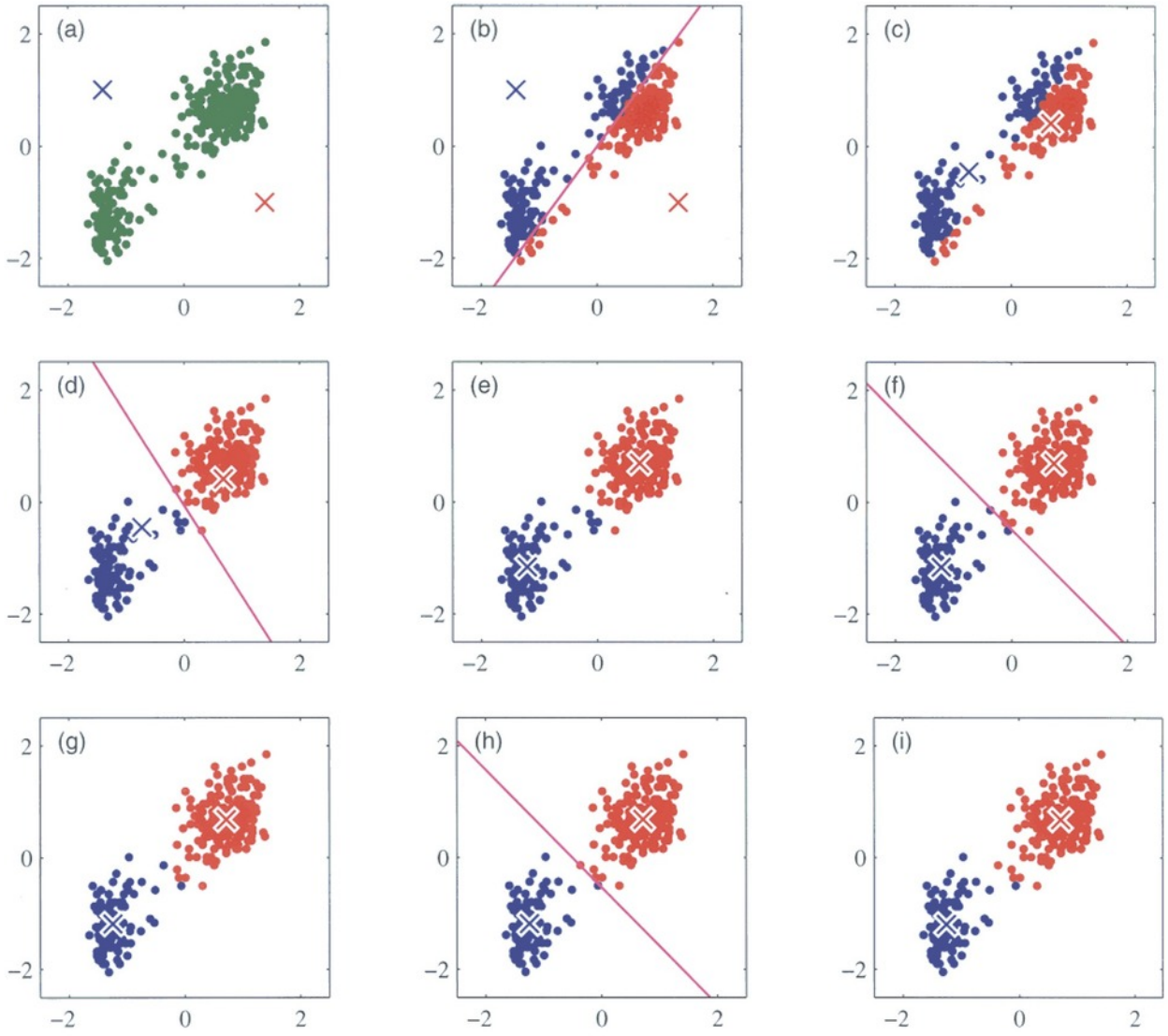

Figure 4: Iterations of K-Means [3]

3 Tuning K-Means

Now that we have understood how K-Means is implemented under the hood, it is worth talking about the hyperparameters that arise with K-Means. There are two main hyperparameters to be concerned with when performing clustering:

[3] http://dendroid.sk/2011/05/09/K-Means-clustering/

1. **Number of Clusters, k**: Because we are working with unlabeled data, we cannot be entirely sure what the right number of clusters should be. Using domain knowledge, we can estimate a range of values for k and then perform cross-validation to determine the right number of clusters.

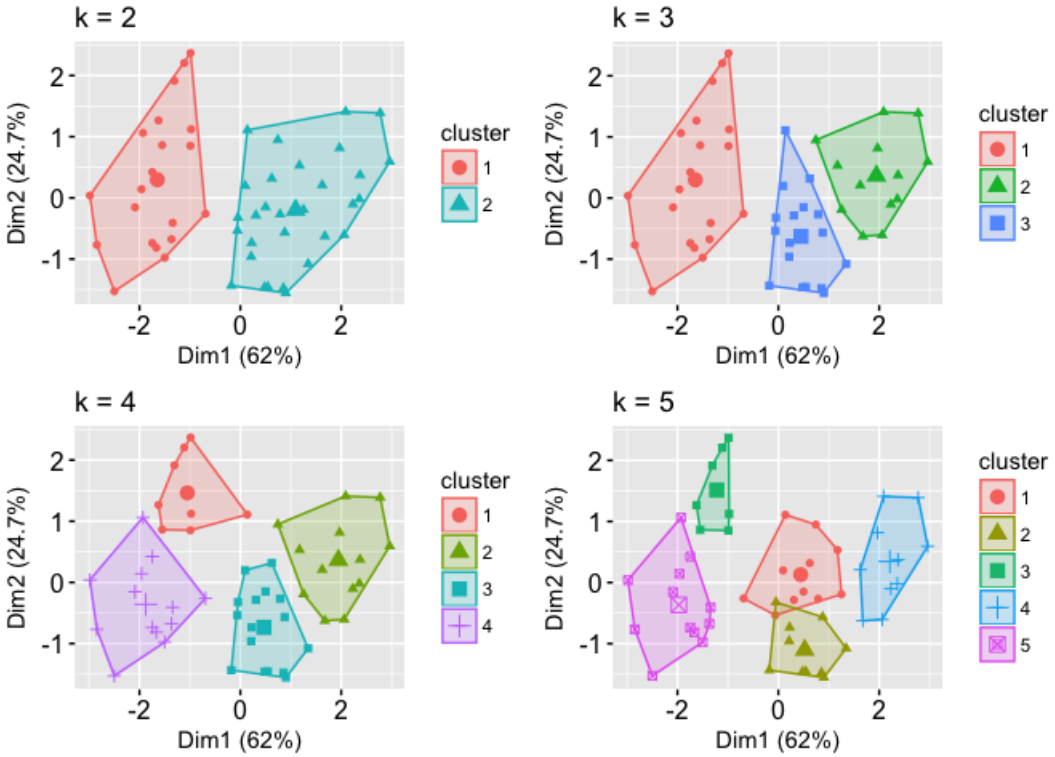

Figure 5: K-Means Clustering under different K [4]

There are a few different methods for choosing k, but we have listed two such popular methods below:

(a) **Elbow Method**

The "elbow" method is commonly used for determining the best number of clusters to use. With K-Means, it's important to note that as the number of clusters increase, the cost is guaranteed to go down because more centroids means that the distance from each point to its closest centroid is guaranteed to decrease. Therefore, we want to pick the number of clusters that best separates the data, while accounting for any noise in the dataset. To prevent overfitting, we pick the cluster number where the cost drops off significantly - the "elbow" of the cost plot.

[4] https://uc-r.github.io/kmeans_clustering

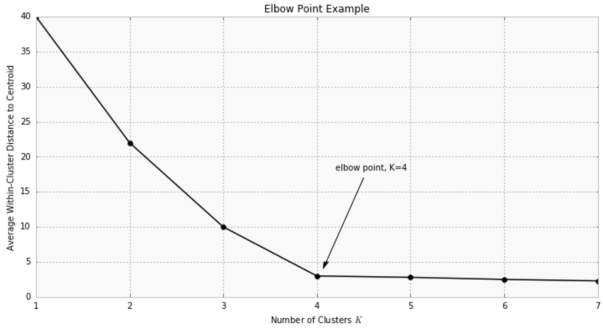

Figure 6: Choosing the right value of K [5]

(b) **Silhouette Method** This method is not used as much as the elbow method because it is more computationally intensive. Essentially, this method id measuring the quality of the clustering by determining how well each object lies within its cluster. The silhouette width, ranging from -1 to 1, is positively correlated with the quality of a clustering assignment. Plot the average silhouette of the observations against different values of k, and pick the k that gives the maximum silhouette value to be the optimal number of clusters. The silouette value is a combination of the similarity of a datapoint to its assigned cluster ($a(i)$) and its dissimilarity to datapoints in other clusters ($b(i)$). To calculate the average silhouette for a single data point, follow the following steps:

Algorithm 2: Silhouette Method

Initialize $s_{total} = 0, n =$number of datapoints;

for *each data point* $i \in C_i$ **do**

$a(i) = \frac{1}{|C_i|-1} \sum_{j \in C_i, i \neq j} d(i,j)$ [a];

$b(i) = \min_{k \neq i} \frac{1}{|C_i|} \sum_{j \in C_k} d(i,j)$;

$s(i) =$

$$\begin{cases} 1 - \frac{a(i)}{b(i)} & a(i) \leq b(i) \\ 0 & a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1 & a(i) \geq b(i) \end{cases}$$

;

$s_{total} += s(i)$

end

silhouette = $\frac{s_{total}}{n}$

[a] d(i, j) is the distance between data points i and j

2. **Dimensionality of data, d**: Initially, we do not have any idea what the clustering will be based off of, so it is difficult for us to determine which features of the data are most important. Some features may be the same across all the data points or have so little variability that they just act as noise. This is why it is sometimes helpful to reduce the dimensionality of the data using techniques like PCA. The right dimensionality of the data can also be selected via cross-validation.

[5] https://towardsdatascience.com/K-Means-clustering-introduction-to-machine-learning-algorithms-c96bf0d5d57a

4 Limitations of K-Means

In the project, you'll have an opportunity to implement and see the K-Means algorithm in action. To understand whether the K-Means algorithm is a good choice for a given machine learning problem, it is important to also understand the limitations of the algorithm. [6]

1. There is no likelihood attached to K-Means, which makes it harder to understand what assumptions we are making on the data. For any given data point, the K-Means algorithm will allow us to assign that point to a cluster. However, we will not have any information about the likelihood or confidence that the assignment is correct.

2. Each feature is treated equally by K-Means. For this reason, in 2-D space, the clusters produced by K-Means are circular. Extending to multidimensional space, the K-Means clusters are spherical in shape. We can also infer this by looking at the sum of squares in the objective function.

3. The K-Means algorithm is based on **hard assignments** - each point belongs to exactly one cluster. However, an algorithm based on **soft assignments** would assign each point to a distribution over the clusters. In this way, not only would we know the cluster that a point is assigned to, but also how far the point is from the other clusters.

4. The results of clustering are highly dependent on our choice of k. This issue can be visualized in the example below. The true assignments of each point are denoted by the different colors. For the choice of k=2, the stars denote the centroids chosen by the K-Means algorithm.

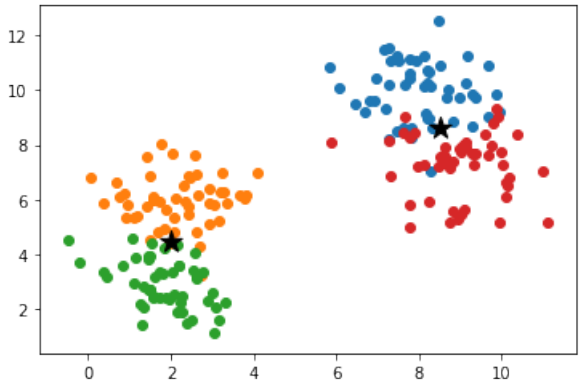

Figure 7: Choosing the right value of K

   To the human eye, k=2 appears to be a reasonable choice for clustering. But in reality, the data is drawn from 4 separate distributions. For this reason, in addition to systematically tuning the value of k, having domain expertise to explain the results of the clustering algorithm can help validate the choice of k.

There are additional shortcomings to K-Means, which you will have the chance to explore in the project! To address some of the limitations to the K-Means algorithm, the next section will highlight a couple variations to the K-Means algorithm.

[6] https://www.eecs189.org/static/notes/n19.pdf

5 Variations of K-Means

We will focus on two of many variations of K-Means: K-medians and kernelized K-Means.

5.1 Variation 1: K-medians

This method is very similar to K-Means, with the alteration being that the median of each cluster is used to determine each centroid rather than the mean. The advantage of this method is that it is much less subject to the impact of outliers. Formally:

Algorithm 3: k-Medians Algorithm

Initialize c_k, k = 1, ..., K;
while *intra-cluster distance keeps changing* **do**
  Given all c_k, update $C_1, ..., C_K$ by assigning each data point $\vec{x}$ to the closest c_k;
  Given the new $C_1, ..., C_K$, update c_k such that each feature is the median of the corresponding features of all data points in C_K;
end

5.2 Variation 2: Kernelized K-Means

Kernelized K-Means is another modification in which the kernel trick is used to measure cluster distance rather than the traditional Euclidean distance. We will examine this in detail during the project.