CS 760: Machine Learning **Spring 2022**

Topic 13: K-Means Clustering

Instructor: Daniel L. Pimentel-Alarcón

13.1 Introduction

Recall that classification can be summarized as assigning a label (class) $\mathrm{y} \in \{1, 2, \ldots, \mathrm{C}\} =: [\mathrm{C}]$ to a data point $\mathbf{x} \in \mathbb{R}^{\mathrm{D}}$ based on a collection of training data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\mathrm{N}} \in \mathbb{R}^{\mathrm{D}}$ whose corresponding classes $\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_{\mathrm{N}} \in [\mathrm{C}]$ are already known. To this end we can use nearest neighbors and other *supervised* learning algorithms.

However, in data science the labels $\{\mathrm{y}_\mathrm{i}\}_{\mathrm{i}=1}^{\mathrm{N}}$ of the training data points $\{\mathbf{x}_\mathrm{i}\}_{\mathrm{i}=1}^{\mathrm{N}}$ are often unavailable:

- Given a collection of vectorized images $\{\mathbf{x}_\mathrm{i}\}_{\mathrm{i}=1}^{\mathrm{N}}$, we want to determine which correspond to the same individuals (but we don't know their names).
- Given a collection of vectors $\{\mathbf{x}_\mathrm{i}\}_{\mathrm{i}=1}^{\mathrm{N}}$ containing information about people's movies ratings (e.g., Netflix or Amazon), we want to determine which people have similar preferences.
- Given a collection of vectors $\{\mathbf{x}_\mathrm{i}\}_{\mathrm{i}=1}^{\mathrm{N}}$ containing genomic sequences from different organisms in a human gut microbiome sample, determine which sequences correspond to the same species.

Unsupervised learning refers to the tasks when labels $\{\mathrm{y}_\mathrm{i}\}_{\mathrm{i}=1}^{\mathrm{N}}$ are unavailable. Clustering is one of such tasks.

13.2 Clustering

The task of clustering can be summarized as splitting a collection of data points into groups such that the points in each group are *similar*. More precisely, given a collection of data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\mathrm{N}} \in \mathbb{R}^{\mathrm{D}}$, we want to identify a partition $\{C_1, C_2, \ldots, C_{\mathrm{K}}\}$ of $[\mathrm{N}]$ (called clusters) such that if $\mathrm{i}, \mathrm{j} \in C_\mathrm{k}$, then $\mathbf{x}_\mathrm{i}$ and $\mathbf{x}_\mathrm{j}$ are *close* to each other (recall that there are several ways to define how *close* two points are, e.g., $\|\mathbf{x}_\mathrm{i} - \mathbf{x}_\mathrm{j}\|_2$, $\|\mathbf{x}_\mathrm{i} - \mathbf{x}_\mathrm{j}\|_1$, or $\langle \mathbf{x}_\mathrm{i}, \mathbf{x}_\mathrm{j} \rangle$).

Recall that norms satisfy the so-called triangle inequality: $\|\mathbf{x}\| + \|\mathbf{y}\| \geq \|\mathbf{x} + \mathbf{y}\|$, which implies that if $\mathbf{x}$ is close to $\mathbf{y}$, and $\mathbf{y}$ is close to $\mathbf{z}$, then $\mathbf{x}$ is also close to $\mathbf{z}$. Using this insight, we can rephrase/adapt our clustering goal in terms of *centers* as follows: given a collection of data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\mathrm{N}} \in \mathbb{R}^{\mathrm{D}}$, we want to identify *centers* $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_{\mathrm{K}} \in \mathbb{R}^{\mathrm{D}}$ that minimize the within-cluster distances:

$$\sum_{\mathrm{k}=1}^{\mathrm{K}} \sum_{\mathrm{i} \in C_\mathrm{k}} \|\boldsymbol{\mu}_\mathrm{k} - \mathbf{x}_\mathrm{i}\|,$$

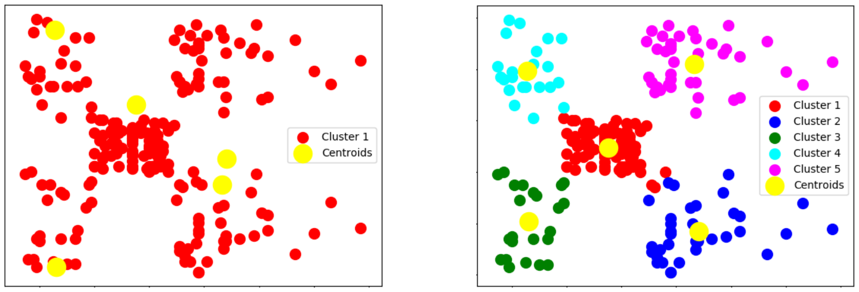

Figure 13.1: Initial and final step of Lloyd's algorithm.

where $\mathrm{i} \in C_{\mathrm{k}}$ if $\|\boldsymbol{\mu}_{\mathrm{k}} - \mathbf{x}_{\mathrm{i}}\| \leq \|\boldsymbol{\mu}_{\ell} - \mathbf{x}\|$ for every $\ell \in [\mathrm{K}]$. Notice that this is a kind of chicken and egg problem: you need to know the clusters $\{C_{\mathrm{k}}\}$ to find the centers $\{\boldsymbol{\mu}_{\mathrm{k}}\}$, and you need to know the centers in order to determine the clusters. This observation is the main insight behind Lloyd's algorithm.

13.3 Lloyd's Algorithm

Lloyd's algorithm, aka the K-means clustering algorithm, is perhaps the simplest unsupervised clustering method, which uses an *alternating* strategy that is very common in machine learning. The main idea is to (i) *pretend* that we know the centers and determine the clusters, (ii) *pretend* that we know the clusters and compute the centers, and then alternate between steps (i) and (ii) until convergence. More precisely, we start with some initial estimates $\hat{\boldsymbol{\mu}}_1, \hat{\boldsymbol{\mu}}_2, \ldots, \hat{\boldsymbol{\mu}}_{\mathrm{K}} \in \mathbb{R}^{\mathrm{D}}$, and then

(i) Assign each datapoint to its closest center to produce a clustering:

$$\hat{C}_{\mathrm{k}} = \Big\{ \mathrm{i} \in [\mathrm{N}] \ : \ \|\hat{\boldsymbol{\mu}}_{\mathrm{k}} - \mathbf{x}_{\mathrm{i}}\| \leq \|\hat{\boldsymbol{\mu}}_{\ell} - \mathbf{x}\| \ \forall \ \ell \in [\mathrm{K}] \Big\}.$$

(ii) Compute the center of each cluster:

$$\hat{\boldsymbol{\mu}}_{\mathrm{k}} = \frac{1}{|\hat{C}_{\mathrm{k}}|} \sum_{\mathrm{i} \in \hat{C}_{\mathrm{k}}} \mathbf{x}_{\mathrm{i}}.$$

Finally, we alternate between steps (i) and (ii) until convergence. See Figure 13.1 to build some intuition.

13.4 Initialization

As with most *alternating* algorithms, Lloyd's algorithm depends heavily on initialization, that is, the choice of initial centers $\{\hat{\boldsymbol{\mu}}_{\mathrm{k}}\}_{\mathrm{k}=1}^{\mathrm{K}}$. There are several popular options:

- **Random samples.** This option simply selects K data points randomly, that is, $\hat{\boldsymbol{\mu}}_{\mathrm{k}} = \mathbf{x}_{\mathrm{i}}$ for some randomly chosen i. This tends to spread out initial centers.
- **Random partition.** This option first partitions data randomly into K clusters, and then computes the initial centers as the mean of each cluster. This tends to place all initial centers close to the center of the entire dataset.

- **K-means++.** This option aims to spread initial centers according to the data distribution. To this end, K-means++ selects one random data point $\mathbf{x}_\mathrm{i}$ as the first center $\hat{\boldsymbol{\mu}}_1$, and then for every $2 \leq \mathrm{k} \leq \mathrm{K}$, it chooses the k^{th} center from the remaining data with probability proportional to its closest existing center. That is, if $\mathbf{x}_\mathrm{j}$ is none of the first $\mathrm{k}-1$ centers, then $\mathbf{x}_\mathrm{j}$ is chosen as the k^{th} center with probability proportional to

$$\min_{\ell \in [\mathrm{k}-1]} \|\hat{\boldsymbol{\mu}}_\ell - \mathbf{x}_\mathrm{j}\|^2.$$

  With this initialization, Lloyd's algorithm is guaranteed to find a solution that is close (within a $\log \mathrm{K}$ factor) to the optimal solution. This is remarkable, because the K-means problem is non-convex, NP-hard, so it is not evident that *any* algorithm should work.