SCHOOL *of* DATA SCIENCE

k-Means Extensions

Distributed Computing
School of Data Science
University of Virginia

Last updated: September 29, 2025

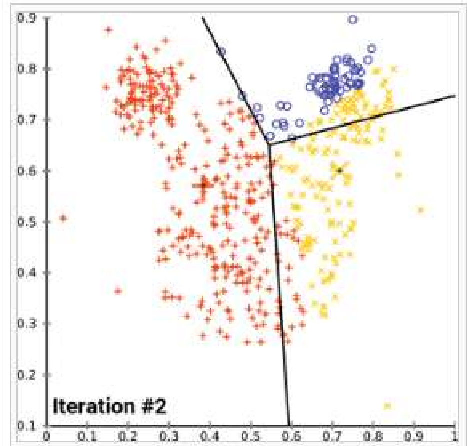
k-means

Given N observations, assign each to k groups

Each group has a *centroid* or balance point

Initialization is essential to result

One method is to randomly assign k points as centroids



Convergence of *k*-means

source: Wikipedia

k-means Limitations

k-means does not perform well in terms of efficiency or quality

Runtime can be exponential

Can be far from global optimum even under repeated random optimization

Better initialization leads to better quality and convergence

k-means++

Choose centers in a controlled fashion

Current set of centers will stochastically bias the choice of next center

The exact algorithm is as follows:

1. Choose one center uniformly at random among the data points.
2. For each data point x not chosen yet, compute D(x), the distance between x and the nearest center that has already been chosen.
3. Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to D(x)^2. This ensures that a very dissimilar point to the previously selected centroid is selected as the next centroid.
4. Repeat Steps 2 and 3 until k centers have been chosen.
5. Now that the initial centers have been chosen, proceed using standard *k*-means clustering.

Source: Wikipedia

k-means++ Limitation

Shortcoming: sequential in nature

Makes k passes over data to produce initial centers

k-means|| Overview

k-means|| Centroids

Let $\mathcal{C} = \{c_1, \ldots, c_k\}$ be a set of points and let $Y \subseteq X$. We define the *cost of* Y with respect to $\mathcal{C}$ as

$$\phi_Y(\mathcal{C}) = \sum_{y \in Y} d^2(y, \mathcal{C}) = \sum_{y \in Y} \min_{i=1,\ldots,k} ||y - c_i||^2.$$

Note: The goal of k-means is to select centroids *C* that minimize cost

k-means|| Algorithm

Algorithm 2 k-means||(k, ℓ) initialization.

1: $\mathcal{C} \leftarrow$ sample a point uniformly at random from X
2: $\psi \leftarrow \phi_X(\mathcal{C})$
3: **for** $O(\log \psi)$ times **do**
4: $\quad \mathcal{C}' \leftarrow$ sample each point $x \in X$ independently with probability $p_x = \frac{\ell \cdot d^2(x, \mathcal{C})}{\phi_X(\mathcal{C})}$
5: $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}'$
6: **end for**
7: For $x \in \mathcal{C}$, set w_x to be the number of points in X closer to x than any other point in $\mathcal{C}$
8: Recluster the weighted points in $\mathcal{C}$ into k clusters

Same as k-means++, select each C to be "far" from other C's

ℓ is an oversampling factor that gives extra centroids

Cull Cs down to size k by weighting and clustering

k-means|| Parallel Implementation

Paper uses MapReduce model for implementation
Many opportunities to parallelize:

Line 4: each mapper can independently sample

Line 7: can divide points among mappers

Calculating cost (sum of squared distances):
Distribute chunks of X to mappers
Mappers compute squared distances
Reducer adds squared distances from mappers

Algorithm 2 k-means|| (k, ℓ) initialization.

1: $\mathcal{C} \leftarrow$ sample a point uniformly at random from X
2: $\psi \leftarrow \phi_X(\mathcal{C})$
3: **for** $O(\log \psi)$ times **do**
4: $\quad \mathcal{C}' \leftarrow$ sample each point $x \in X$ independently with probability $p_x = \frac{\ell \cdot d^2(x, \mathcal{C})}{\phi_X(\mathcal{C})}$
5: $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}'$
6: **end for**
7: For $x \in \mathcal{C}$, set w_x to be the number of points in X closer to x than any other point in $\mathcal{C}$
8: Recluster the weighted points in $\mathcal{C}$ into k clusters