L9 Assignment-Based Clustering

Monday, February 17, 2020 5:03 PM

Input: 1) $X \subset \mathbb{R}^d$

- data point $x_i \in X$

2) Distance $\underset{\text{metric}}{d}: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

Goal: $S = \{S_1, S_2, S_3, \ldots S_k\}$

- $S_i \subset X$
- $S_i \cap S_j = \emptyset$
- $\cup S_i = X$

Assignment-Based Clustering

- Clusters $S_1, S_2, S_3, \ldots, S_k$
  - each with a "centroid"

    $C = \{c_1, c_2, \ldots, c_k\}$

    restricts the type of clusters being considered

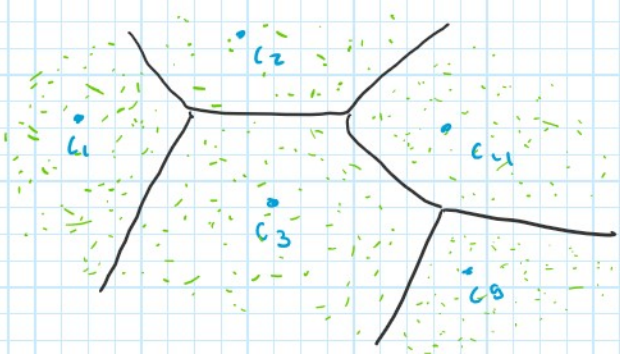
    ↳ representation points

Nearest Neighbor Function

$$\phi_C : \mathbb{R}^d \to C$$

- In particular...

$$\phi_C(x) = \underset{c_i \in C}{\operatorname{argmin}}\, d(x, c_i)$$



Goal: Find $C = \{c_1, c_2, \ldots c_k\}$ given some k.

Formulations:

- k-means: minimize $\sum_{x \in X} d(x, \phi_C(x))^2$ ← most popular

  Lloyds $\quad d =$ Euclidean

- k-center: minimizes $\max_{x \in X} d(x, \phi_C(x))$

  Gonzalez

- k-median: minimize $\sum d(x, \phi_C(x))$

- k-mediod: minimize $\sum_{C \subset X} d(x, \phi_C(x))$

Gonzalez Algorithm For K-Center

Main Idea: Build G incrementally.

$G_1 \rightarrow G_2 \rightarrow G_3 \rightarrow \ldots \rightarrow G_K = G$

ie, build these in K steps where $|G_i| = i$

The Algorithm:

S1. Choose some random c_1 arbitrarily $\Rightarrow G_1 = \{c_1\}$

S2. for $j = 2$ to K:

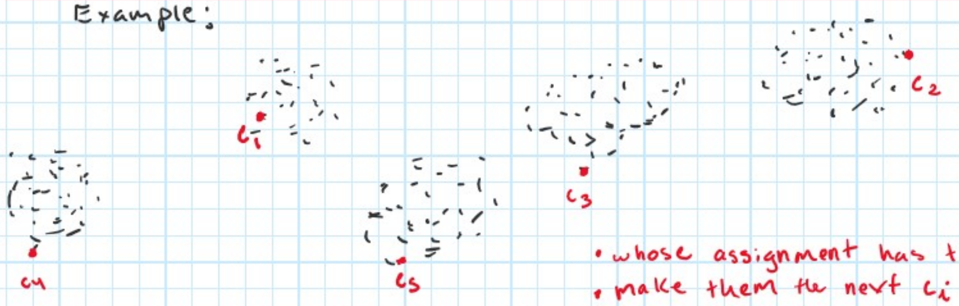
set $c_j = \arg\max_{x \in X} d(x, \phi_{G_{j-1}}(x))$

ie, the distance between x and its assigned max!

if d is a metric, then:

↳ output 2-approximation of the optimal.

Example:



O(Kn) time
- on each round (K of them)
- check re-assignment to cluster
- keep track of the max
  - the max becomes the new center

- whose assignment has the greatest cost?
- make them the next c_i

- implementation tip:
  - maintain an array of size equal to the # of points
  - for each point, store the index of the closest center seen so far.
  - update the array
  - This done in linear time each round, where there are K rounds.

Lloyd's Algorithm For K-Means: (Minimizing the Sum of squared errors)

S1. choose K points $\rightarrow G$ $\qquad d =$ euclidean

S2. repeat

S2a) For all $x \in X$, find $\phi_G(x) \rightarrow S_1, S_2, \ldots, S_K$ where each $S_i = \{x \in X \mid \phi_G(x) = c_i\}$

S2b) For all $i \in 1, \ldots, K$, let $c_i = \text{average}(S_i)$

S3. until (S unchanged or change is small)

* usually < 20 steps

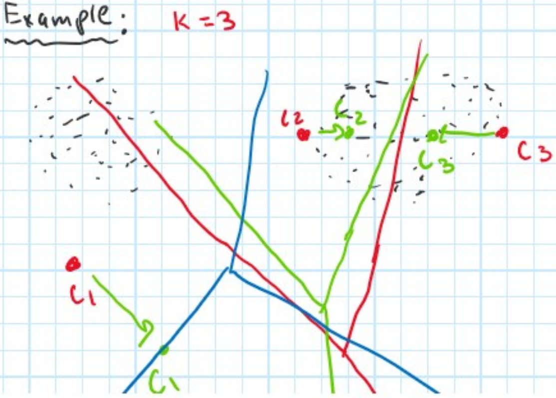
$c_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$

- add up all values of x assigned to the center
- divide the number of points

$\uparrow$ $\arg\min_{z \in \mathbb{R}^d} \sum_{x \in S_i} \|z - x\|^2$

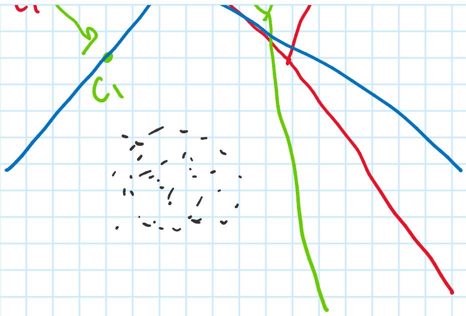
(the point which minimizes the squared errors)

Example: $K = 3$



- re-assign centers (cost ↓ each round)
- re-assign the points (cost ↓ each round)

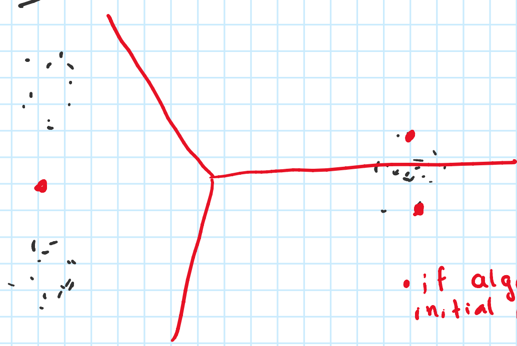
↳ thus, we are converging



↳ thus, we are converging

But Loyd's can get stuck at a local min

ex



• if algo gets stuck, just start fresh with new initial points.

How do we choose the initial k centers?

1. Pick random subset $C \subset X$ (random indices from the set of x)
   - issue: small clusters are unlikely to receive a centroid
     ↳ the algorithm will only divide the center, which is sub-optimal

2. Use the Gonzalez Algorithm to initialize points
   - issue: sensitive to outliers. Also deterministic.

3. k-means++

   S1. choose c_1 arbitrarily where $c_1 \in X$

   S2. for $i = 2$ to k:

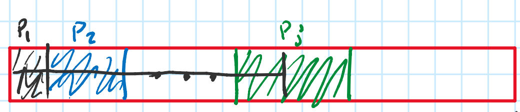
   Choose c_i from x w/probability proportional to

   $$v_j = d(x, \phi_{C_{i-1}}(x_j))^2$$

   ie, instead of picking longest distance, we will calculate all the distances, square them, and then sample from the datapoints were we're more likely to pick the points with a larger squared difference.

$$V = \sum_{j=1}^{n} v_j \Rightarrow prob(x_j) = \frac{v_j}{V} = p_j$$

Most implement to implement k-means++



$0.55 = v \in Unif(0,1)$

$(\sum p_j = 1)$

(each time we pick a center, we generate a probability distribution and pick an element from it)