K-means algorithm

Oliver W. Layton

CS251: Data analysis and visualization

Lecture 23, Fall 2018

Friday April 5

Plan

- K-means algorithm
- Leader algorithm

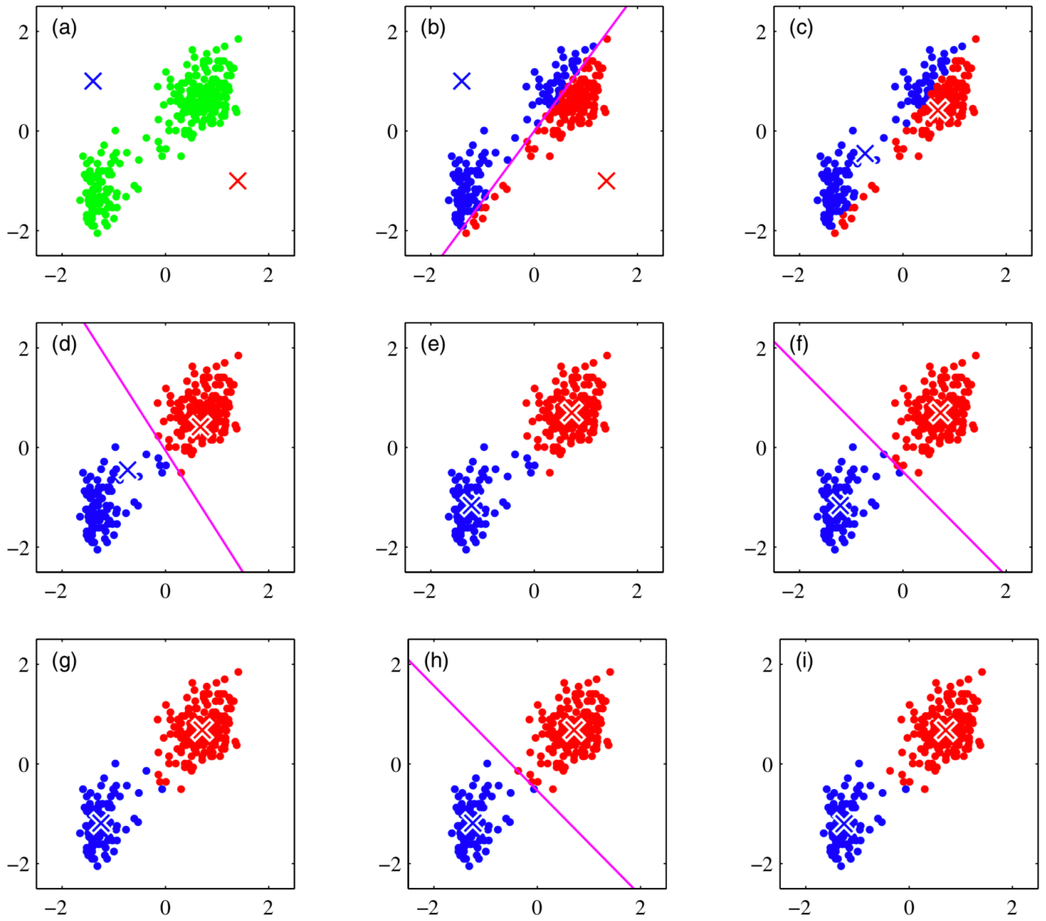
K-means

- The most popular algorithm for determining clusters is called **K-means**.
- *Goal*: Divvy up data into K *distinct* clusters to minimize the L^2 distance of group members from the K cluster centers.
- Cluster "center" is the mean of all points assigned to that cluster (**centroid**).

K-means algorithm

- Initially, pick K centroids, points in data space (number of dimensions == number of features).
  - Uniform random in feature space OR pick K data points randomly.
- **Assign step:** Assign each point to the closest centroid.
- **Update step:** Update centroids to be the mean of data points belonging to each respective cluster.
- Repeat until centroids shift less than some threshold amount.
- Show short Keynote animation

K-means step-by-step



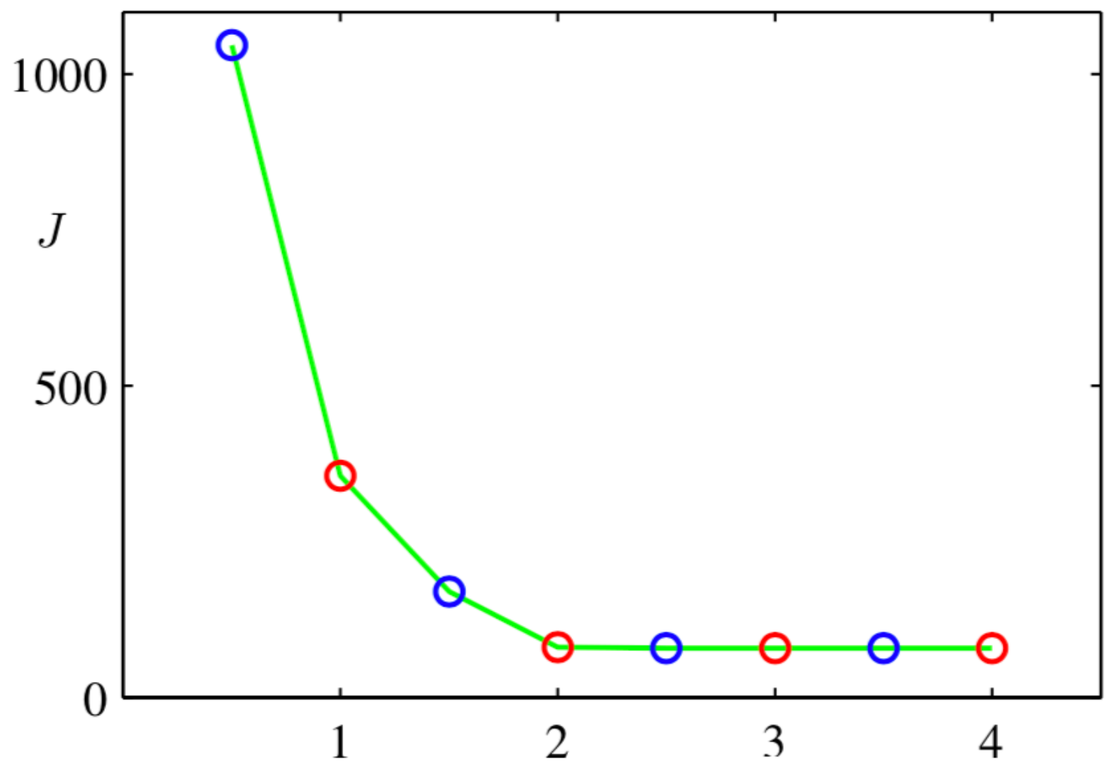
K-means objective

Goal: Minimize the TOTAL Euclidean distance (J) from ALL points to the point that defines membership to their cluster. This is often called the **sum of squared errors (SSE)**.

$$SSE = \sum_{j=0}^{C-1} \sum_{i=0}^{M-1} ||x_i - c_j||_2^2$$

For C cluster centroids and M data points.

Convergence of K-means as a function of iteration number



Data from two slides back.

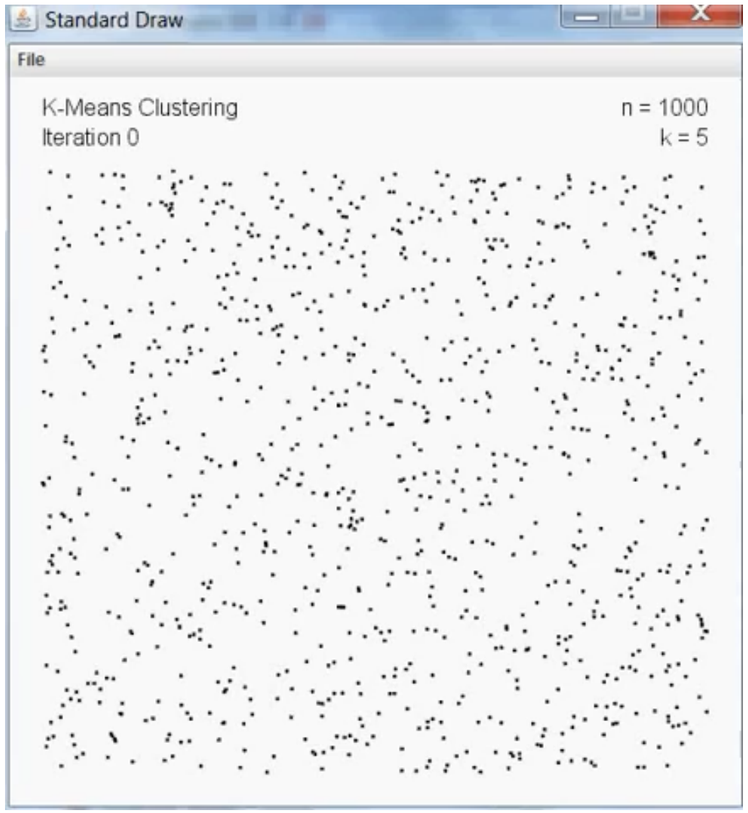
K-means is a heuristic

- To assign N data points to K clusters, there are $S(N, K) = \frac{1}{K!} \sum_{k=1}^{K} (-1)^{K-k} \binom{K}{k} k^N$ options!
- 4 clusters, 19 data points: 10^{10} possible assignments — what a nightmare!!
- *There is no a definitive K-means algorithm*. Most options are **heuristic algorithms** that slice through the vast possible clustering space favoring speed using "rules of thumb". They may find "good" solutions; they cannot promise to provide the "best" way to cluster any dataset.
- "The" K-means algorithm defined a few slides back is called **Lloyd's algorithm**. It is a local, greedy search algorithm.
  - It iteratively assigns data point-by-point d_i to the group with the closest centroid.

Problems with K-means?

1. *Parameter selection*: How do we pick K? What is the optimal value?
    - Let's run K-means on random data, varying K.
    - We will discuss methods to determine #clusters in a dataset very soon.
2. *Distance metric*: How do we define "close" when assigning to centroids?
3. *Sensitivity to initial conditions*: Clustering accuracy and runtime depends on good initial centroids.

K-means results can vary quite a bit



Video credit: https://www.youtube.com/watch?v=BVFG7fd1H30

Improving K-means initialization

- Uniform random selection (points or space) is bad (why?).
- The **K-means++** weighting spreads out initial K centroids by assigning low probabilities around existing centroids.
- Probability of picking one of the K centroids follows a V-shape distribution.

K-means++ algorithm

- Pick a data point to be centroid 1 according to uniform random distribution.
- Pick next centroid $k \leq K$, weighting each data point x_i probability $p(x_i) = \frac{D(x_i)^2}{\sum_j D(x_j)^2}$, where $D(x)$ defines the shortest distance to any existing centroid.
- Repeat this step until we have K centroids. Run K-means like usual.

Quickly cluster a dataset without specifying K

- The **Leader algorithm** quickly detects clusters without specifying number of clusters K.
- Instead, we divvy up M data points into clusters of radius ≤ T.
- Definitions
  - **1st Leader L(1)**: 1st data point.
  - **Threshold T**: maximum distance from 1st leader to assign data points to the 1st leader's group.

Leader algorithm

- Put all data points into cluster 1 until an data point has distance from L(1) > T.
- That data points becomes L(2).
- If possible, put next data point to cluster 1, otherwise cluster 2, otherwise it becomes L(3), etc.
- Benefit: Very efficient. Only 1 pass thru data.