Clustering and PCA Recitation

Nupur Chatterji, Kenny Marino, Colin White
Clustering

Unsupervised Learning - unlabeled data

- Automatically organize data
- Understand structure in data
- Preprocessing for further analysis
Applications (Clustering comes up everywhere...)

- Cluster news articles or web pages or search results by topic.

- Cluster protein sequences by function or genes according to expression profile.

- Cluster users of social networks by interest (community detection).
Objective Based Clustering

**Input:** A set $S$ of $n$ points, also a *distance/dissimilarity measure* specifying the distance $d(x,y)$ between pairs $(x,y)$.

E.g., # keywords in common, edit distance, wavelets coef., etc.

**Goal:** output a partition of the data.

- **k-means:** find center pts $c_1, c_2, ..., c_k$ to minimize $\sum_{i=1}^{n} \min_{j\in\{1,...,k\}} d^2(x^i, c_j)$

- **k-median:** find center pts $c_1, c_2, ..., c_k$ to minimize $\sum_{i=1}^{n} \min_{j\in\{1,...,k\}} d(x^i, c_j)$

- **K-center:** find partition to minimize the maximum radius
Euclidean k-means Clustering

- **Input:** a set of \( n \) points, \( x^1, x^2, \ldots, x^n \), in \( \mathbb{R}^d \), an integer \( k \)
- **Output:** \( k \) “centers” \( c_1, c_2, \ldots, c_k \)

Try to minimize the distance from each point \( x^i \) to its closest center

\[
\sum_{i=1}^{n} \min_{j \in \{1, \ldots, k\}} \left\| x^i - c_j \right\|^2
\]
K-means complexity

- Hard to solve even when $k=2$ and $d=2$
- $k=1$ is easy to solve
- $d=1$ is easy to solve (dynamic programming)
Common Heuristic in Practice: The Lloyd's method


**Input:** A set of $n$ datapoints $x^1, x^2, \ldots, x^n$ in $\mathbb{R}^d$

**Initialize** centers $c_1, c_2, \ldots, c_k \in \mathbb{R}^d$ and clusters $C_1, C_2, \ldots, C_k$ in any way.

**Repeat** until there is no further change in the cost.

- For each $j$: $C_j \leftarrow \{x \in S \text{ whose closest center is } c_j\}$
- For each $j$: $c_j \leftarrow \text{mean of } C_j$

Holding $c_1, c_2, \ldots, c_k$ fixed, pick optimal $C_1, C_2, \ldots, C_k$

Holding $C_1, C_2, \ldots, C_k$ fixed, pick optimal $c_1, c_2, \ldots, c_k$
Lloyd’s initialization

Initialization is very important for Lloyd’s method

- Random initialization
- Farthest-first traversal: iteratively choose farthest point from current set
- $d^2$-sampling (k-means++) iteratively choose a point $v$ with probability $d_{\min}(v,C)^2$, where $C$ is the list of current centers

**Theorem:** k-means++ always attains an $O(\log k)$ approximation to the optimal k-means solution in expectation.
K-means runtime

- K-means++ initialization $O(nkd)$ time
- Lloyd’s method: $O(nkd)$ time

Exponential number of rounds in the worst case

Small number of rounds in practice

Expected number of rounds is polynomial time under *smoothed analysis*
Hierarchical Clustering

- What if we don’t know the right value of k? (often the case)

Often leads to natural solutions
Bottom-Up (agglomerative)

Have a distance measure on pairs of objects.
\[ d(x, y) - \text{distance between } x \text{ and } y \]

E.g., \# keywords in common, edit distance, etc

- Single linkage:
  \[ \text{dist}(C, C') = \min_{x \in C, x' \in C'} \text{dist}(x, x') \]

- Complete linkage:
  \[ \text{dist}(C, C') = \max_{x \in C, x' \in C'} \text{dist}(x, x') \]

- Average linkage:
  \[ \text{dist}(C, C') = \frac{\text{avg}}{x \in C, x' \in C'} \text{dist}(x, x') \]
Single Linkage

Bottom-up (agglomerative)
- Start with every point in its own cluster.
- Repeatedly merge the “closest” two clusters.

Single linkage: $\text{dist}(C, C') = \min_{x \in C, x' \in C'} \text{dist}(x, x')$

Dendogram

Runtime:
- $O(n^3)$ is easy
- Can achieve $O(n^2 \log n)$