Clustering.
Unsupervised Learning

Maria-Florina Balcan
04/08/2019
Clustering, Informal Goals

**Goal:** Automatically partition *unlabeled* data into groups of similar datapoints.

**Question:** When and why would we want to do this?

**Useful for:**
- Automatically organizing data.
- Understanding hidden structure in data.
- Preprocessing for further analysis.
  - Representing high-dimensional data in a low-dimensional space (e.g., for visualization purposes).
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).
Applications (Clustering comes up everywhere...)

- Cluster customers according to purchase history.

- Cluster galaxies or nearby stars (e.g. Sloan Digital Sky Survey)

- And many many more applications....
Clustering

Today:

• Objective based clustering
  • K-means clustering
• Hierarchical clustering
Objectives 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 $\sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} d^2(x^i, c_j)$
- **k-median:** find center pts $c_1, c_2, ..., c_k$ to minimize $\sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} d(x^i, c_j)$
- **K-center:** find partition to minimize the maximum radius
Euclidean k-means Clustering

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

target #clusters $k$

**Output:** $k$ representatives $c_1, c_2, \ldots, c_k \in \mathbb{R}^d$

**Objective:** choose $c_1, c_2, \ldots, c_k \in \mathbb{R}^d$ to minimize

$$\sum_{i=1}^{n} \min_{j \in \{1, \ldots, k\}} \left\| \mathbf{x}^i - c_j \right\|^2$$
Euclidean k-means Clustering

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

**target #clusters $k$**

**Output:** $k$ representatives $c_1, c_2, \ldots, c_k \in \mathbb{R}^d$

**Objective:** choose $c_1, c_2, \ldots, c_k \in \mathbb{R}^d$ to minimize

$$\sum_{i=1}^{n} \min_{j\in\{1,\ldots,k\}} \left\| x^i - c_j \right\|^2$$

Natural assignment: each point assigned to its closest center, leads to a Voronoi partition.
Euclidean k-means Clustering

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

target #clusters $k$

**Output:** $k$ representatives $c_1, c_2, \ldots, c_k \in \mathbb{R}^d$

**Objective:** choose $c_1, c_2, \ldots, c_k \in \mathbb{R}^d$ to minimize

$$\sum_{i=1}^{n} \min_{j \in \{1, \ldots, k\}} \left\| \mathbf{x}^i - c_j \right\|^2$$

**Computational complexity:**

NP hard: even for $k = 2$ [Dagupta'08] or $d = 2$ [Mahajan-Nimbhorkar-Varadarajan09]

There are a couple of easy cases...
An Easy Case for k-means: k=1

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

Output: $c \in \mathbb{R}^d$ to minimize $\sum_{i=1}^{n} ||x^i - c||^2$

Solution: The optimal choice is $\mu = \frac{1}{n} \sum_{i=1}^{n} x^i$

Idea: bias/variance like decomposition

$$\frac{1}{n} \sum_{i=1}^{n} ||x^i - c||^2 = ||\mu - c||^2 + \frac{1}{n} \sum_{i=1}^{n} ||x^i - \mu||^2$$

Avg k-means cost wrt $c$  Avg k-means cost wrt $\mu$

So, the optimal choice for $c$ is $\mu$. 
Another Easy Case for k-means: d=1

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

**Output:** $c \in \mathbb{R}^d$ to minimize $\sum_{i=1}^{n} \|x^i - c\|^2$

Extra-credit homework question

Hint: dynamic programming in time $O(n^2k)$. 


Common Heuristic in Practice:
The Lloyd’s method


**Input:** A set of $n$ datapoints $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$ 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 $j$: $c_j \leftarrow \text{mean of } C_j$
Common Heuristic in Practice: The Lloyd's method


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

Initialize centers $c_1, c_2, ..., c_k \in \mathbb{R}^d$ and clusters $C_1, C_2, ..., C_k$ 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 $c_1, c_2, ..., c_k$ fixed, pick optimal $C_1, C_2, ..., C_k$

Holding $C_1, C_2, ..., C_k$ fixed, pick optimal $c_1, c_2, ..., c_k$
Common Heuristic: The Lloyd’s method

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

**Initialize** centers $c_1, c_2, ..., c_k \in \mathbb{R}^d$ and clusters $C_1, C_2, ..., C_k$ 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 $j$: $c_j \leftarrow \text{mean of } C_j$

**Note:** it always converges.

- the cost always drops and
- there is only a finite $\#s$ of Voronoi partitions
  (so a finite $\#$ of values the cost could take)
Initialization for the Lloyd’s method

**Input:** A set of \( n \) datapoints \( \mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^n \) in \( \mathbb{R}^d \)

**Initialize centers** \( \mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_k \in \mathbb{R}^d \) and clusters \( C_1, C_2, ..., C_k \) in any way.

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

- For each \( j \): \( C_j \leftarrow \{ x \in S \text{ whose closest center is } \mathbf{c}_j \} \)
- For each \( j \): \( \mathbf{c}_j \leftarrow \text{mean of } C_j \)

- **Initialization is crucial** (how fast it converges, quality of solution output)
- **Discuss techniques commonly used in practice**
  - Random centers from the datapoints (repeat a few times)
  - Furthest traversal
  - K-means ++ (works well and has provable guarantees)
Lloyd’s method: Random Initialization
Lloyd’s method: Random Initialization

Example: Given a set of datapoints
Lloyd’s method: Random Initialization

Select initial centers at random
Lloyd’s method: Random Initialization

Assign each point to its nearest center
Lloyd’s method: Random Initialization

Recompute optimal centers given a fixed clustering
Lloyd’s method: Random Initialization

Assign each point to its nearest center
Lloyd’s method: Random Initialization

Recompute optimal centers given a fixed clustering
Lloyd’s method: Random Initialization

Assign each point to its nearest center
Lloyd’s method: Random Initialization

Recompute optimal centers given a fixed clustering

Get a good quality solution in this example.
It always converges, but it may converge at a local optimum that is different from the global optimum, and in fact could be arbitrarily worse in terms of its score.
Lloyd’s method: Performance

Local optimum: every point is assigned to its nearest center and every center is the mean value of its points.
Lloyd’s method: Performance

It is arbitrarily worse than optimum solution....
Lloyd’s method: Performance

This bad performance, can happen even with well separated Gaussian clusters.
Lloyd’s method: Performance

This bad performance can happen even with well separated Gaussian clusters.

Some Gaussian are combined....
Lloyd’s method: Performance

- If we do random initialization, as $k$ increases, it becomes more likely we won’t have perfectly picked one center per Gaussian in our initialization (so Lloyd’s method will output a bad solution).

  - For $k$ equal-sized Gaussians, $\Pr[\text{each initial center is in a different Gaussian}] \approx \frac{k!}{k^k} \approx \frac{1}{e^k}$

- Becomes unlikely as $k$ gets large.
Another Initialization Idea: Furthest Point Heuristic

Choose \( c_1 \) arbitrarily (or at random).

- For \( j = 2, \ldots, k \)
  - Pick \( c_j \) among datapoints \( x^1, x^2, \ldots, x^n \) that is farthest from previously chosen \( c_1, c_2, \ldots, c_{j-1} \)

Fixes the Gaussian problem. But it can be thrown off by outliers....
Furthest point heuristic does well on previous example
Furthest point initialization heuristic sensitive to outliers

Assume $k=3$
Furthest point initialization heuristic sensitive to outliers

Assume $k=3$
K-means++ Initialization: $D^2$ sampling [AV07]

- Interpolate between random and furthest point initialization.
- Let $D(x)$ be the distance between a point $x$ and its nearest center. Chose the next center proportional to $D^2(x)$.

Choose $c_1$ at random.

For $j = 2, ..., k$
- Pick $c_j$ among $x^1, x^2, ..., x^n$ according to the distribution.

$$\Pr(c_j = x^i) \propto \min_{j' < j} \left\| x^i - c_{j'} \right\|^2 \frac{1}{D^2(x^i)}$$

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

Running Lloyd's can only further improve the cost.
K-means++ Idea: $D^2$ sampling

- Interpolate between random and furthest point initialization
- Let $D(x)$ be the distance between a point $x$ and its nearest center. Choose the next center proportional to $D^\alpha(x)$.
  
  - $\alpha = 0$, random sampling
  - $\alpha = \infty$, furthest point (Side note: it actually works well for k-center)
  - $\alpha = 2$, k-means++

Side note: $\alpha = 1$, works well for k-median
**K-means++/ Lloyd’s Running Time**

- **K-means ++ initialization:** $O(nd)$ and one pass over data to select next center. So $O(nkd)$ time in total.

- **Lloyd’s method**

  Repeat until there is no change in the cost.
  - For each $j$: $C_{j} \leftarrow \{x \in S \text{ whose closest center is } c_{j}\}$
  - For each $j$: $c_{j} \leftarrow \text{mean of } C_{j}$

  Each round takes time $O(nkd)$.

- Exponential # of rounds in the worst case [AV07].

- Expected polynomial time in the smoothed analysis (non worst-case) model!
K-means++/ Lloyd’s Summary

- K-means++ always attains an $O(\log k)$ approximation to optimal k-means solution in expectation.
- Running Lloyd’s can only further improve the cost.
- Exponential # of rounds in the worst case [AV07].
- Expected polynomial time in the smoothed analysis model!
- Does well in practice.
What value of k???

- Heuristic: Find large gap between $k$-1-means cost and $k$-means cost.

- Hold-out validation/cross-validation on auxiliary task (e.g., supervised learning task).

- Try hierarchical clustering.
• A hierarchy might be more natural.
• Different users might care about different levels of granularity or even prunings.
What You Should Know

• Partitional Clustering. k-means and k-means ++
  • Lloyd’s method
  • Initialization techniques (random, furthest traversal, k-means++)