# $k$-means Clustering

**TOP: Data Clustering 076/091**

Instructor: Sayan Bandyapadhyay

Portland State University

# Outline

# Real Points

Suppose the set of points $X$ are from $\mathbb{R}^d$

- A natural center of points is the average point or mean

$$\mu = \frac{1}{|S|} \cdot \sum_{x \in S} x$$

- Here the sum is coordinate-wise total:
  $(1, 3) + (2, 5) = (3, 8)$

# Real Points

Suppose the set of points $X$ are from $\mathbb{R}^d$

- A natural center of points is the average point or mean

$$\mu = \frac{1}{|S|} \cdot \sum_{x \in S} x$$

- Here the sum is coordinate-wise total:
  $(1, 3) + (2, 5) = (3, 8)$
- This is the basis of the $k$-means algorithm
- Proposed by Lloyd in 1957, published in 1982
- Also by Max in 1960

# Real Points

Suppose the set of points $X$ are from $\mathbb{R}^d$

- A natural center of points is the average point or mean

$$\mu = \frac{1}{|S|} \cdot \sum_{x \in S} x$$

- Here the sum is coordinate-wise total:
  $(1, 3) + (2, 5) = (3, 8)$
- This is the basis of the $k$-means algorithm
- Proposed by Lloyd in 1957, published in 1982
- Also by Max in 1960

Euclidean distance of $x$ and $y$, $||x - y|| = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$

# The $k$-means Algorithm (Lloyd-Max)

**Algorithm** $k$-means

**Require:** Set of points $X$

Start with centers $c_1, \ldots, c_k$ chosen arbitrarily from $X$
**repeat**
    **for** each point $x_i \in X$ **do**
        Assign $x_i$ to cluster $C_j$ that minimizes $||x_i - c_j||$
    **end for**
    **for** each cluster $C_j$ **do**
        $c_j \leftarrow \frac{1}{|C_j|} \cdot \sum_{x_i \in C_j} x_i$
    **end for**
**until** cluster centers do not change

# Time Complexity of $k$-means

**Algorithm** $k$-means

**Require:** Set of points $X$

Start with centers $c_1, \ldots, c_k$ chosen arbitrarily from $X$
**repeat**
    **for** each point $x_i \in X$ **do**
        Assign $x_i$ to cluster $C_j$ that minimizes $||x_i - c_j||$
    **end for**
    **for** each cluster $C_j$ **do**
        $c_j \leftarrow \frac{1}{|C_j|} \cdot \sum_{x_i \in C_j} x_i$
    **end for**
**until** cluster centers do not change

# Time Complexity of $k$-means

**Algorithm** $k$-means

**Require:** Set of points $X$

1: Start with centers $c_1, \ldots, c_k$ chosen arbitrarily from $X$
2: **repeat**
3: **for** each point $x_i \in X$ **do**
4: Assign $x_i$ to cluster $C_j$ that minimizes $||x_i - c_j||$
5: **end for**
6: **for** each cluster $C_j$ **do**
7: $c_j \leftarrow \frac{1}{|C_j|} \cdot \sum_{x_i \in C_j} x_i$
8: **end for**
9: **until** cluster centers do not change

- Again we need a "rate of cost decrease" type argument as for $k$-median
- What is a suitable cost function that the mean minimizes?

# A Suitable Cost Function

- For what function $g(.,.)$, mean$(S)$ minimizes $\sum_{x \in S} g(x, c)$ over all $c$?

# A Suitable Cost Function

- For what function $g(.,.)$, mean$(S)$ minimizes $\sum_{x \in S} g(x, c)$ over all $c$?
- Such $g$ is called Bregman divergence that encompasses many functions

# A Suitable Cost Function

- For what function $g(.,.)$, mean$(S)$ minimizes $\sum_{x \in S} g(x, c)$ over all $c$?
- Such $g$ is called Bregman divergence that encompasses many functions
- One such $g$ is squared Euclidean distance

$$g(x, c) = ||x - c||^2$$

# A Suitable Cost Function

- For what function $g(.,.)$, mean$(S)$ minimizes $\sum_{x \in S} g(x, c)$ over all $c$?
- Such $g$ is called Bregman divergence that encompasses many functions
- One such $g$ is squared Euclidean distance

$$g(x, c) = ||x - c||^2$$

- This leads to our $k$-means clustering problem for real points with Euclidean distance

# $k$-means clustering

Given a set $X$ of $n$ points in the metric space $(\mathcal{U}, d)$

- Find a set $C$ of $k$ points (cluster centers) in $\mathcal{U}$ that minimizes,

$$\mathrm{cost}(C) = \sum_{p \in X} d(p, NearestCenter(p))^2$$

# Euclidean $k$-means clustering

Given a set $X$ of $n$ points in $\mathbb{R}^d$

- Find a set $C$ of $k$ points (cluster centers) in $\mathbb{R}^d$ that minimizes,

$$\text{cost}(C) = \sum_{p \in X} ||p - NearestCenter(p)||^2$$

# Time Complexity of Lloyd's Algorithm

- $M_1, M_2, \ldots, M_\ell$ are the sets of means computed over $\ell$ iterations

To show: $\mathrm{cost}(M_\ell) < \mathrm{cost}(M_{\ell-1}) < \mathrm{cost}(M_{\ell-2}) < \ldots < \mathrm{cost}(M_1)$

# Time Complexity of Lloyd's Algorithm

- $M_1, M_2, \ldots, M_\ell$ are the sets of means computed over $\ell$ iterations

To show: $\text{cost}(M_\ell) < \text{cost}(M_{\ell-1}) < \text{cost}(M_{\ell-2}) < \ldots < \text{cost}(M_1)$

- In every iteration, means are picked as centers of the clusters
- A mean minimizes the sum-of-squares cost function
- So, $k$-means cost also decreases for the new set of centers

# Time Complexity

- In every iteration, cost decreases
- The algorithm never cycles – The same set of centers never comes back
- Number of iterations is bounded by the number of distinct sets of means

# Time Complexity

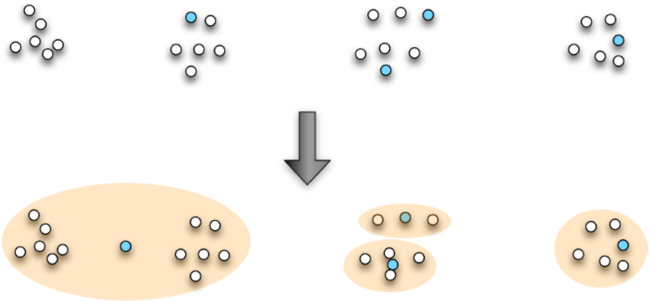
- In every iteration, cost decreases
- The algorithm never cycles – The same set of centers never comes back
- Number of iterations is bounded by the number of distinct sets of means
- $2^n$ subsets: $2^n$ distinct means; $(2^n)^k$ distinct sets of means

# Time Complexity

- In every iteration, cost decreases
- The algorithm never cycles – The same set of centers never comes back
- Number of iterations is bounded by the number of distinct sets of means
- $2^n$ subsets: $2^n$ distinct means; $(2^n)^k$ distinct sets of means
- Lloyd's algorithm always terminates
- In practice, it is very fast
- One can also terminate the algorithm after a few iterations

# Outline

# Analysis of Quality



Initialization/Seeding is the key

# Initialization

- Does random initialization help?

# Initialization

- Does random initialization help? No! We still can get nearby centers

# Initialization

- Does random initialization help? No! We still can get nearby centers
- We need well-separated centers – can we use Greedy 2 - Furthest point algorithm for $k$-center?

# Initialization

- Does random initialization help? No! We still can get nearby centers
- We need well-separated centers – can we use Greedy 2 - Furthest point algorithm for $k$-center?
- Sensitive to outliers

# Initialization

- Does random initialization help? No! We still can get nearby centers
- We need well-separated centers – can we use Greedy 2 - Furthest point algorithm for $k$-center?
- Sensitive to outliers
- We need something in between

# Initialization

- Does random initialization help? No! We still can get nearby centers
- We need well-separated centers – can we use Greedy 2 - Furthest point algorithm for $k$-center?
- Sensitive to outliers
- We need something in between
- we should pick far away points only if there are many points in the vicinity

# Initialization

- Does random initialization help? No! We still can get nearby centers
- We need well-separated centers – can we use Greedy 2 - Furthest point algorithm for $k$-center?
- Sensitive to outliers
- We need something in between
- we should pick far away points only if there are many points in the vicinity
- We should not pick an outlier as a center

# Initialization

- Does random initialization help? No! We still can get nearby centers
- We need well-separated centers – can we use Greedy 2 - Furthest point algorithm for $k$-center?
- Sensitive to outliers
- We need something in between
- we should pick far away points only if there are many points in the vicinity
- We should not pick an outlier as a center

This leads to a new seeding algorithm!

# Non-uniformly Random Seeding

- Start with a uniformly random center

# Non-uniformly Random Seeding

- Start with a uniformly random center
- Next center is chosen from a distribution biased towards far away points

# Non-uniformly Random Seeding

- Start with a uniformly random center
- Next center is chosen from a distribution biased towards far away points
- Cost of a point $x_i$, $\mathrm{cost}(x_i, C) = \min_{c \in C} ||x_i - c||^2$

# Non-uniformly Random Seeding

- Start with a uniformly random center
- Next center is chosen from a distribution biased towards far away points
- Cost of a point $x_i$, $\text{cost}(x_i, C) = \min_{c \in C} ||x_i - c||^2$
- Define the probability $p_i = \text{cost}(x_i, C)/\text{cost}(C)$

# The $k$-means++ Algorithm

**Algorithm** $k$-means++

**Require:** Set of points $X$, parameter $k$

Select $c_1$ randomly from $X$
$C \leftarrow \{c_1\}$
**while** $(|C| \neq k)$ **do**
  **for** each $i = 1$ to $n$ **do**
    $\text{cost}(x_i, C) \leftarrow \min_{c \in C} ||x_i - c||^2$
  **end for**
  $\text{cost}(C) \leftarrow \sum_{i=1}^{n} \text{cost}(x_i, C)$
  Sample a random point $y \in X$, selecting each $x_i$ w.p. $p_i = \text{cost}(x_i, C)/\text{cost}(C)$
  $C \leftarrow C \cup \{y\}$
**end while**
Invoke Lloyd's algorithm with $C$ as the seed

# Analysis of $k$-means++

- Time complexity: $O(nk) +$ Lloyd's

# Analysis of $k$-means++

- Time complexity: $O(nk) +$ Lloyd's
- Approximation factor: $O(\log k)$

# Analysis of $k$-means++

- Time complexity: $O(nk) +$ Lloyd's
- Approximation factor: $O(\log k)$

Compare this with $p$-swap Local search

- $O(n^{p+1}k^{p+1}\log n)$ time, but $9 + (1/p)$-approximation
- Works in general metric space (even for non-numerical data)