CS685: Data Mining
Partitioning-Based Clustering Methods

Arnab Bhattacharya
arnabb@cse.iitk.ac.in

Computer Science and Engineering,
Indian Institute of Technology, Kanpur
http://web.cse.iitk.ac.in/~cs685/

1st semester, 2020-21

Mon 1030-1200 (online)

Partitioning-Based Clustering

- Suppose number of clusters is k and number of data points is n
- Objective is to minimize the sum-squared error (SSE)
- Error for each point is its distance from the corresponding cluster centre

$$SSE = \sum_{j=1}^{k} \sum_{O_i \in C_j} (c_j - O_i)^2$$

- O_i are the points
- c_j are the cluster centres of the clusters C_j
- How to find the *optimal* clusters (i.e., the optimal partitioning) that minimize the SSE?

Partitioning-Based Clustering

- Suppose number of clusters is k and number of data points is n
- Objective is to minimize the sum-squared error (SSE)
- Error for each point is its distance from the corresponding cluster centre

$$SSE = \sum_{j=1}^{k} \sum_{O_i \in C_j} (c_j - O_i)^2$$

- O_i are the points
- c_j are the cluster centres of the clusters C_j
- How to find the *optimal* clusters (i.e., the optimal partitioning) that minimize the SSE?
- All possible choices
- Impractical as running time is $O(n^k)$

K-Means

- K-means is a very efficient *heuristic*
- Iterative algorithm

K-Means

- K-means is a very efficient *heuristic*
- Iterative algorithm
- Start with k *random* cluster centers
- Cluster centers are *not* necessarily actual data points

K-Means

- K-means is a very efficient *heuristic*
- Iterative algorithm
- Start with k *random* cluster centers
- Cluster centers are *not* necessarily actual data points
- Two sub-steps in each iteration
  - Assign each point to the nearest cluster center
  - Update cluster center as *mean* of points assigned to it

K-Means

- K-means is a very efficient *heuristic*
- Iterative algorithm
- Start with k *random* cluster centers
- Cluster centers are *not* necessarily actual data points
- Two sub-steps in each iteration
  - Assign each point to the nearest cluster center
  - Update cluster center as *mean* of points assigned to it
- Continue till
  - There is no change of cluster assignment for *any* point
  - Number of iterations reaches a threshold

K-Means

- K-means is a very efficient *heuristic*
- Iterative algorithm
- Start with k *random* cluster centers
- Cluster centers are *not* necessarily actual data points
- Two sub-steps in each iteration
  - Assign each point to the nearest cluster center
  - Update cluster center as *mean* of points assigned to it
- Continue till
  - There is no change of cluster assignment for *any* point
  - Number of iterations reaches a threshold
- In practice, start with multiple random centers

K-Means

- K-means is a very efficient *heuristic*
- Iterative algorithm
- Start with k *random* cluster centers
- Cluster centers are *not* necessarily actual data points
- Two sub-steps in each iteration
  - Assign each point to the nearest cluster center
  - Update cluster center as *mean* of points assigned to it
- Continue till
  - There is no change of cluster assignment for *any* point
  - Number of iterations reaches a threshold
- In practice, start with multiple random centers
- If nominal data, update cluster center as *most frequent* value in the cluster, i.e., the *mode*
- Then, it is called k-modes

K-Means

- K-means is a very efficient *heuristic*
- Iterative algorithm
- Start with k *random* cluster centers
- Cluster centers are *not* necessarily actual data points
- Two sub-steps in each iteration
  - Assign each point to the nearest cluster center
  - Update cluster center as *mean* of points assigned to it
- Continue till
  - There is no change of cluster assignment for *any* point
  - Number of iterations reaches a threshold
- In practice, start with multiple random centers
- If nominal data, update cluster center as *most frequent* value in the cluster, i.e., the *mode*
- Then, it is called k-modes
- Converges to *local minimum*

K-Means

- K-means is a very efficient *heuristic*
- Iterative algorithm
- Start with k *random* cluster centers
- Cluster centers are *not* necessarily actual data points
- Two sub-steps in each iteration
  - Assign each point to the nearest cluster center
  - Update cluster center as *mean* of points assigned to it
- Continue till
  - There is no change of cluster assignment for *any* point
  - Number of iterations reaches a threshold
- In practice, start with multiple random centers
- If nominal data, update cluster center as *most frequent* value in the cluster, i.e., the *mode*
- Then, it is called k-modes
- Converges to *local minimum*
- Running time for t iterations is $O(n.k.t)$

Why Mean as Cluster Centre?

- For each cluster, *mean* is chosen as the cluster centre

Why Mean as Cluster Centre?

- For each cluster, *mean* is chosen as the cluster centre
- Minimizing SSE with respect to cluster centres

$$\frac{\partial}{\partial c_j} SSE = \frac{\partial}{\partial c_j} \sum_{j=1}^{k} \sum_{O_i \in C_j} (c_j - O_i)^2$$

$$= \sum_{j=1}^{k} \sum_{O_i \in C_j} \frac{\partial}{\partial c_j} (c_j - O_i)^2 = \sum_{j=1}^{k} \sum_{O_i \in C_j} 2(c_j - O_i)$$

- For each cluster

$$\sum_{O_i \in C_j} 2(c_j - O_i) = 0 \implies |C_j|.c_j = \sum_{O_i \in C_j} O_i$$

$$\implies c_j = \sum_{O_i \in C_j} O_i / |C_j| = mean_{O_i \in C_j} \{O_i\}$$

How to Choose k?

- Domain knowledge is the best

How to Choose k?

- Domain knowledge is the best
- Visual inspection of clusters for low dimensional datasets

How to Choose k?

- Domain knowledge is the best
- Visual inspection of clusters for low dimensional datasets
- Choosing the one with minimum squared error is not appropriate

How to Choose k?

- Domain knowledge is the best
- Visual inspection of clusters for low dimensional datasets
- Choosing the one with minimum squared error is not appropriate
- When k increases, clusters becomes tighter, and squared error decreases
- When $k = n$, squared error is 0

How to Choose k?

- Domain knowledge is the best
- Visual inspection of clusters for low dimensional datasets
- Choosing the one with minimum squared error is not appropriate
- When k increases, clusters becomes tighter, and squared error decreases
- When $k = n$, squared error is 0
- Silhouette index or Silhouette coefficient

K-Medoids

- Minimizes the sum of absolute errors (SAE) and not squared errors

$$SAE = \sum_{j=1}^{k} \sum_{O_i \in C_j} |c_j - O_i|$$

- O_i are the points
- c_j are the cluster centres of the clusters C_j
- Therefore, k-medoids uses *medians* instead of means

K-Medoids

- Minimizes the sum of absolute errors (SAE) and not squared errors

$$SAE = \sum_{j=1}^{k} \sum_{O_i \in C_j} |c_j - O_i|$$

- O_i are the points
- c_j are the cluster centres of the clusters C_j
- Therefore, k-medoids uses *medians* instead of means
- Chooses *actual* data objects as cluster centers

K-Medoids

- Minimizes the sum of absolute errors (SAE) and not squared errors

$$SAE = \sum_{j=1}^{k} \sum_{O_i \in C_j} |c_j - O_i|$$

- O_i are the points
- c_j are the cluster centres of the clusters C_j
- Therefore, k-medoids uses *medians* instead of means
- Chooses *actual* data objects as cluster centers
- Three methods
  - Partitioning Around Medoids (PAM)
  - Clustering Large Applications (CLARA)
  - Clustering Large Applications based upon Randomized Search (CLARANS)

Why Median as Cluster Centre?

- For each cluster, the *median* is chosen as the cluster centre

Why Median as Cluster Centre?

- For each cluster, the *median* is chosen as the cluster centre
- Minimizing SAE with respect to cluster centres

$$\frac{\partial}{\partial c_j} SAE = \frac{\partial}{\partial c_j} \sum_{j=1}^{k} \sum_{O_i \in C_j} |c_j - O_i|$$

$$= \sum_{j=1}^{k} \sum_{O_i \in C_j} \frac{\partial}{\partial c_j} |c_j - O_i| = \sum_{j=1}^{k} \sum_{O_i \in C_j} sign(c_j - O_i)$$

- For each cluster

$$\sum_{O_i \in C_j} sign(c_j - O_i) = 0$$

$$\implies c_j = median_{O_i \in C_j} \{O_i\}$$

K-Medoids Algorithm

- Two steps: build and swap

K-Medoids Algorithm

- Two steps: build and swap
- Build
  - Select k objects as centres randomly
  - These are *representative objects* or medoids
  - For every other object, select the nearest cluster centre and assign it to that cluster

K-Medoids Algorithm

- Two steps: build and swap
- Build
  - Select k objects as centres randomly
  - These are *representative objects* or medoids
  - For every other object, select the nearest cluster centre and assign it to that cluster
- Swap
  - Examine *pair* of objects, one a medoid and another a non-medoid
  - *Swap* their status, i.e., make the non-medoid a medoid and the medoid a non-medoid
  - If quality of clustering improves, retain the swap; otherwise, discard it
- Continue iterating till no improvement

K-Medoids Algorithm

- Two steps: build and swap
- Build
  - Select k objects as centres randomly
  - These are *representative objects* or medoids
  - For every other object, select the nearest cluster centre and assign it to that cluster
- Swap
  - Examine *pair* of objects, one a medoid and another a non-medoid
  - *Swap* their status, i.e., make the non-medoid a medoid and the medoid a non-medoid
  - If quality of clustering improves, retain the swap; otherwise, discard it
- Continue iterating till no improvement
- Quality of clustering is measured by SAE
- *Cost* of swap is defined as the change in total SAE for a swap
  - Swap may initiate change in cluster for some objects

K-Medoids Algorithm

- Two steps: build and swap
- Build
  - Select k objects as centres randomly
  - These are *representative objects* or medoids
  - For every other object, select the nearest cluster centre and assign it to that cluster
- Swap
  - Examine *pair* of objects, one a medoid and another a non-medoid
  - *Swap* their status, i.e., make the non-medoid a medoid and the medoid a non-medoid
  - If quality of clustering improves, retain the swap; otherwise, discard it
- Continue iterating till no improvement
- Quality of clustering is measured by SAE
- *Cost* of swap is defined as the change in total SAE for a swap
  - Swap may initiate change in cluster for some objects
- PAM, CLARA and CLARANS differ on how to swap

PAM

- Partitioning Around Medoids (PAM)
- Swap a medoid with *every* non-medoid
- Basically examine all possible swaps

PAM

- Partitioning Around Medoids (PAM)
- Swap a medoid with *every* non-medoid
- Basically examine all possible swaps
- Robust to noise since it is exhaustive

PAM

- Partitioning Around Medoids (PAM)
- Swap a medoid with *every* non-medoid
- Basically examine all possible swaps
- Robust to noise since it is exhaustive
- Time taken for examining cost of a swap is $O(n - k)$
- There are a total of $O(k(n - k))$ swaps
- So, total time complexity is $O(k(n - k)^2) \approx O(n^2)$
- Too impractical

PAM

- Partitioning Around Medoids (PAM)
- Swap a medoid with *every* non-medoid
- Basically examine all possible swaps
- Robust to noise since it is exhaustive
- Time taken for examining cost of a swap is $O(n - k)$
- There are a total of $O(k(n - k))$ swaps
- So, total time complexity is $O(k(n - k)^2) \approx O(n^2)$
- Too impractical
- Sampling to reduce time

CLARA

- Clustering LARge Applications (CLARA)
- Create a random *sample* of the data
- Perform PAM on the sample, i.e., k medoids chosen only from the sample
- Cost of swap is measured on *all* non-medoids

CLARA

- Clustering LARge Applications (CLARA)
- Create a random *sample* of the data
- Perform PAM on the sample, i.e., k medoids chosen only from the sample
- Cost of swap is measured on *all* non-medoids
- Perform random sampling multiple times

CLARA

- Clustering LARge Applications (CLARA)
- Create a random *sample* of the data
- Perform PAM on the sample, i.e., k medoids chosen only from the sample
- Cost of swap is measured on *all* non-medoids
- Perform random sampling multiple times
- If sample size is s, cost of running PAM is $O(k(s-k)^2)$
- Time taken for assigning a non-medoid to a cluster is $O(k)$
- Time taken for assigning all non-medoids is $O(k(n-k))$
- So, total time complexity is $O(k(s-k)^2 + k(n-k)) \approx O(s^2 + n)$

CLARA

- Clustering LARge Applications (CLARA)
- Create a random *sample* of the data
- Perform PAM on the sample, i.e., k medoids chosen only from the sample
- Cost of swap is measured on *all* non-medoids
- Perform random sampling multiple times
- If sample size is s, cost of running PAM is $O(k(s-k)^2)$
- Time taken for assigning a non-medoid to a cluster is $O(k)$
- Time taken for assigning all non-medoids is $O(k(n-k))$
- So, total time complexity is $O(k(s-k)^2 + k(n-k)) \approx O(s^2 + n)$
- Efficient
- Performs well if sampling is good

CLARA

- Clustering LARge Applications (CLARA)
- Create a random *sample* of the data
- Perform PAM on the sample, i.e., k medoids chosen only from the sample
- Cost of swap is measured on *all* non-medoids
- Perform random sampling multiple times
- If sample size is s, cost of running PAM is $O(k(s-k)^2)$
- Time taken for assigning a non-medoid to a cluster is $O(k)$
- Time taken for assigning all non-medoids is $O(k(n-k))$
- So, total time complexity is $O(k(s-k)^2 + k(n-k)) \approx O(s^2 + n)$
- Efficient
- Performs well if sampling is good
- May miss out on good cluster centres due to sampling

CLARANS

- Clustering LARge Applications based upon raNdomized Search (CLARANS)
- Instead of creating samples, look at entire data
- However, swap a medoid with a non-medoid only a fixed number of times
- Also, run for at most a fixed number of iterations

CLARANS

- Clustering LARge Applications based upon raNdomized Search (CLARANS)
- Instead of creating samples, look at entire data
- However, swap a medoid with a non-medoid only a fixed number of times
- Also, run for at most a fixed number of iterations
- Better than CLARA as it looks at entire data
- Faster than PAM as it does not examine all non-medoids
- Experimental results show that CLARANS obtains the best clusters

Graph-Based Framework

- Graph of *swaps*
- Each node represents a particular set of k medoids
- Total of

Graph-Based Framework

- Graph of *swaps*
- Each node represents a particular set of k medoids
- Total of $\binom{n}{k}$ nodes
- Node u is connected to node v if they differ by *exactly* one medoid
  - An edge denotes a possible swap
- Cost of edge is change in SAE
- Each node has

Graph-Based Framework

- Graph of *swaps*
- Each node represents a particular set of k medoids
- Total of $\binom{n}{k}$ nodes
- Node u is connected to node v if they differ by *exactly* one medoid
  - An edge denotes a possible swap
- Cost of edge is change in SAE
- Each node has $k(n-k)$ neighbors

Graph-Based Framework

- Graph of *swaps*
- Each node represents a particular set of k medoids
- Total of $\binom{n}{k}$ nodes
- Node u is connected to node v if they differ by *exactly* one medoid
  - An edge denotes a possible swap
- Cost of edge is change in SAE
- Each node has $k(n-k)$ neighbors
- PAM searches for the best node in the entire graph
- From a node, it searches all its neighbors and moves to the one with the lowest SAE

Graph-Based Framework

- Graph of *swaps*
- Each node represents a particular set of k medoids
- Total of $\binom{n}{k}$ nodes
- Node u is connected to node v if they differ by *exactly* one medoid
  - An edge denotes a possible swap
- Cost of edge is change in SAE
- Each node has $k(n-k)$ neighbors
- PAM searches for the best node in the entire graph
- From a node, it searches all its neighbors and moves to the one with the lowest SAE
- CLARA creates a similar graph but with only s data objects

Graph-Based Framework

- Graph of *swaps*
- Each node represents a particular set of k medoids
- Total of $\binom{n}{k}$ nodes
- Node u is connected to node v if they differ by *exactly* one medoid
  - An edge denotes a possible swap
- Cost of edge is change in SAE
- Each node has $k(n-k)$ neighbors
- PAM searches for the best node in the entire graph
- From a node, it searches all its neighbors and moves to the one with the lowest SAE
- CLARA creates a similar graph but with only s data objects
- CLARANS uses the entire graph
- It examines only a fixed number of neighbors
- It also moves for only a limited number of times