Graph Clustering Algorithms

Report

on

Algorithms

by

Kishore Kumar Kalathur Chenchu (M21CS058)
Prabhala Sandhya Gayatri (M21CS060)
Tejaswee A (M21CS064)

Course Instructor

Dr. Anand Mishra
Assistant Professor
Department of Computer Science & Engineering
Indian Institute of Technology Jodhpur

1 K-Means

Algorithm 1: K-Means

Input :
$D = d\_1, d\_2, d\_3, ..., d\_n$ //n-data points
k //desired number of clusters

Output :
k-clusters

Time Complexity :
$O(k * n * t)$
where,
n is the number of samples
t is the number of iterations
k is the number of clusters

Steps :
1. Initialize k cluster centroids randomly from D which are the initial centroids.
2.**while** *convergence is not met* **do**
    For every i, set
    $$c^{(i)} := \underset{j}{\operatorname{argmin}} ||x^{(i)} - \mu\_j||^2$$
    For each j, set
    $$\mu\_j := \frac{\sum\_i=1^m 1\{c^{(i)}=j\}x^{(i)}}{\sum\_i=1^m 1\{c^{(i)}=j\}}$$
end

Example

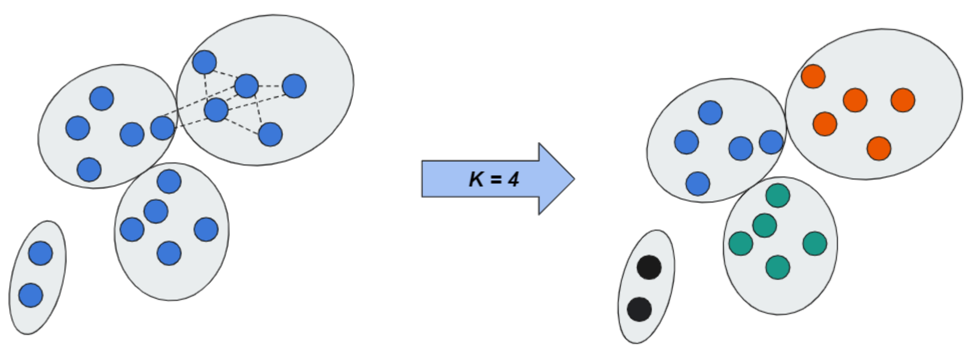


Figure 1: K-Means Example

2 K-Center

Algorithm 2: Greedy K-center Approximation

Input :

Undirected Complete Graph $G(V, E)$ with distance d_i_j>=0 between each pair of vertices i, j ∈V

k //number of centers

Output :

k-clusters

Steps :

Greedy-Algorithm(G, k)

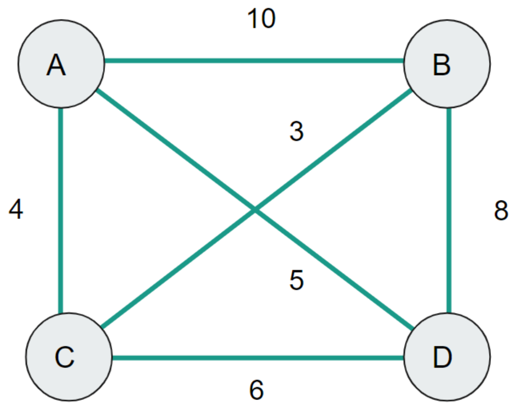
1. Choosing the first center randomly.
2. Choose remaining $(k - 1)$ centers using the criteria:

   Let $c\_1, c\_2, c\_3, ..., c\_i$ be the already chosen centers.

   Choose $(i + 1)^{th}$ center by picking the node which is farthest from the already selected centers which means that node p has the value as maximum min$[dist(p, c_1), dist(p, c_2), dist(p, c_3), ...dist(p, c_n)]$

Example

Let us consider a hostel block where the rooms are distance (in meters) apart as shown in the graph below. The authorities decide to establish Wi-Fi connection with routers placed at specific distances so that every occupant is equally benefited. Our aim is to find minimum numbers of routers to be installed.



Given *k* = 2, the most optimal solution is to place the routers near rooms *C* and *D* where the maximum distance becomes 6.

Figure 2: K-Center Example

3 K-Medians

Algorithm 3: K-Medians

Input :
$D = d\_1, d\_2, d\_3, ..., d\_n$ //n-data points
k //desired number of clusters

Output :
k-clusters

Time Complexity :
$O(k * n^2 * t)$
where,
n is the number of samples
t is the number of iterations
k is the number of clusters

Steps :
1. Convergence condition is $\sum\_k = 1^k \sum\_d\_i \in C\_k|d\_ij - median\_kj|$
2. Select k points as the initial k medians
3.**while** *convergence is not met* **do**
a)Assign every point to its nearest median.
b)Recompute the median using the median of each individual point.
end

Example

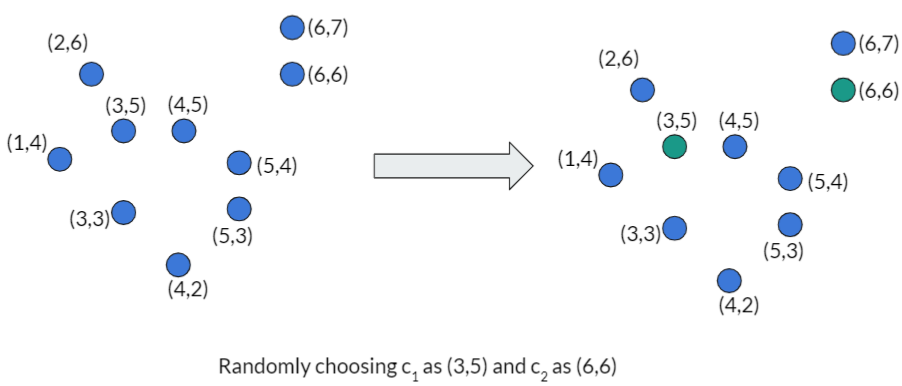


Figure 3: K-Medians Example

4 Greedy Agglomeration

Algorithm 4: Greedy Agglomeration

Input :
Let $X = x\_1, x\_2, x\_3, ..., x\_n$ be set of data points. //n-data points

Output :
k-clusters

Time Complexity :
$O(k * n^2)$
where,
n is the number of samples or elements to be clustered
k is the number of clusters

Steps :

1. Begin with the disjoint clustering having level $L(0) = 0$ and sequence number $m = 0$.
2. Find the least distance pair of clusters, $d[(r),(s)] = minimum(d[(i),(j)])$
3. Increment the sequence number: $m = m + 1$.
   Merge clusters (r) and (s) into a single cluster to form the next clustering m.
   Set the level of this clustering to $L(m) = d[(r),(s)]$.
4. Update the distance matrix D.
   New cluster = (r, s)
   Old cluster(k) = $d[(k),(r,s)] = min(d[(k),(r)], d[(k),(s)])$.
5. Stop when a single cluster containing all the data points is obtained, else repeat from step 2.

Example

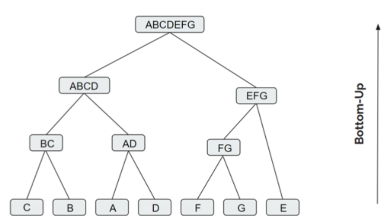


Figure 4: Greedy Agglomeration Example

5 Markov clustering Algorithm

Algorithm 5: Markov clustering Algorithm

Input : Undirected graph,power parameter e, and inflation parameter r.

Output : number of clusters

Time Complexity :
$O(n * k^2)$
where,
n is the number of nodes in the graph
k is the number of clusters

Steps :
1. $A := A + I$ //Add self-loops to the graph where I is identity matrix
2. $M := AD^{-1}$ //Initialize M as the canonical transition matrix i.e. Normalizing the matrix

repeat
  $M := M\_exp := Expand(M)$ //Expand the matrix by taking e^{th} power of the matrix
  $M := M\_inf := Inflate(M, r)$ //Inflation parameter r controls the extent of strengthening or weakening.
  $M := Prune(M)$ //Rounding the values

until *M converges*;
Here interpreting resulting Matrix M to discover clusters.

Example

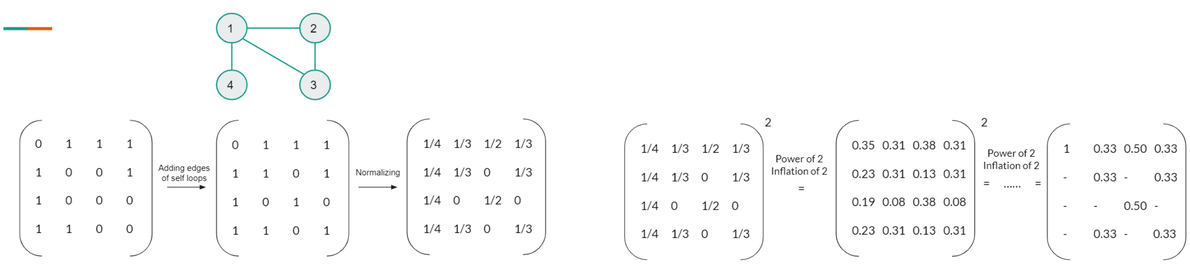


Figure 5: MCL Example

6 Multilevel Clustering

Algorithm 6: Multilevel Clustering Algorithm

Input : graph, coarsener, refiner, reduction factor

Output : clustering

Steps :

```
// coarsening phase
level[1] ← graph;
repeat
    clustering← vertices of level[l];
    clustering← coarsener(level[l],clustering,reduction factor);
    if cluster count reduced then
        level[l + 1] ← contract each cluster of clustering into a single vertex;
    end
until cluster count not reduced;
// refinement phase
clustering← vertices of level[l_max];
for l from l_max − 1 to 1 do
    clustering← project clustering from level[l + 1] to level[l];
    clustering← refiner(level[l], clustering);
end
```

7 Clique Percolation Method

Algorithm 7: Clique Percolation Method

Input : Complex Graph G

Output : Candidate community set C_L

Steps :

1. UF $\leftarrow$ Union Find data structure
2. Dict $\leftarrow$ Empty Dictionary

for *each* ***k*** *clique* $c\_k \in G$ **do**
  S $\leftarrow \phi$
  for *each* $(k-1)$*clique* $c\_k - 1 \subset c\_k$ **do**
    if $c\_k - 1 \in Dict.keys()$ **then**
      $P \leftarrow$ *UF.Find(Dict[*$c\_k - 1$*])*
    end
    else
      $P \leftarrow$ *UF.MakeSet()*
      Dict[$c\_k - 1$*]* $\leftarrow p$
    end
    $S \leftarrow S \cup \{p\}$
  end
  UF.Union(S)
end

Example

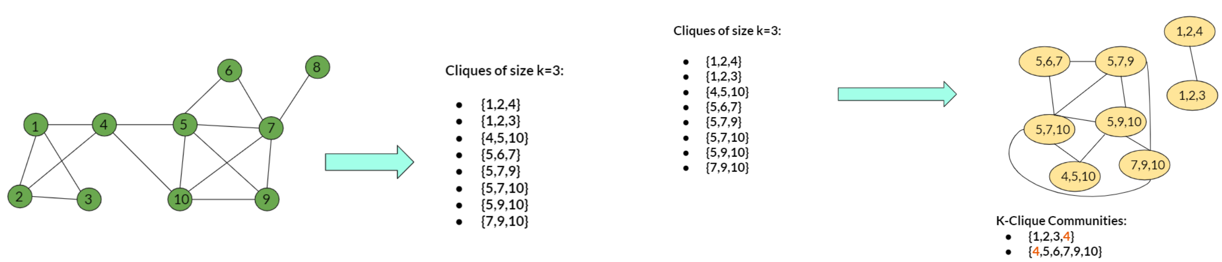


Figure 6: Clique Percolation Method Example