Tutorial

Emad Zahedi, Vahid Mirjalili

Abstract

Here we briefly talk about the K-means clustering algorithm.

1. k-means

Let we are given a data $X = \{x_1, x_2, \cdots, x_n\}$, where $x_i \in R^d$, and we want to partition the data into k disjoint clusters $C_1, \cdots, C_k$. The $K-$means algorithm is designed in such a way that every cluster is associated with a centroid (center) and defined as follows:

....

```
1: Select $k$ data points as initial centroids.
2: **Repeat**
3: Form $k$ clusters by assigning each vertex to the closest centroid.
4: recompute the centroids for all clusters.
5: **Until** centroids do not change.
```

....

This method is an iterative approach for minimizing sum of squared error, that is the sum of the Euclidean distances between every data point and it's associated cluster. In other word, mathematically the error is given by

$$SSE = \sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij}||x_i - c_j||_2^2,$$

where $w_{ij} = 1$ if x_i is in cluster C_j; otherwise $w_{ij} = 0$, hence $\sum_j w_{ij} = 1$, and c_j is the representative center for cluster C_j. To minimize SSE we consider two cases based on variables c_j and w_{ij}, if we fix a c_j then

$$w_{ij} = \begin{cases} 1, & \text{if } j = \text{argmin}_x \, ||x_i - c_j||_2 \\ 0, & \text{otherwise.} \end{cases}$$

And if w_{ij} is fixed, then define

$$L = \sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij}||x_i - c_j||_2^2 - \sum_{i=1}^{n} \lambda_i \Bigg(\sum_{j=1}^{k} w_{ij} - 1 \Bigg),$$

Email addresses: j.e.zahedi@gmail.com (Emad Zahedi), v.mirjalili@gmail.com (Vahid Mirjalili) as by getting derivative of L respect with c_j we have

$$\frac{\partial L}{\partial c_j} = -2\sum_{i=1}^{n} w_{ij}(x_i - c_j),$$

and solving for c_j when $\frac{\partial L}{\partial c_j} = 0$, gives

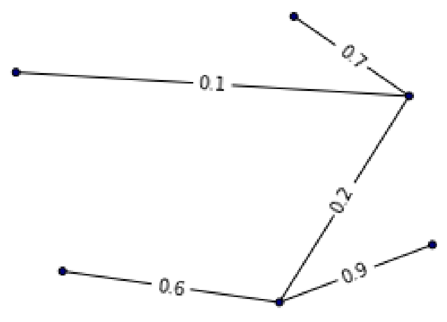
$$c_j = \frac{\sum_{i=1}^{n} w_{ij}x_i}{\sum_{i=1}^{n} w_{ij}}.$$

The $K-$means algorithm is fast and it has time complexity $O(n*k*I*d)$ where n is number of points, k is number of clusters, I is number of iterations, and d is number of attributes[]. The main disadvantages of this algorithm would be the final solution is related to the the initial position of the cluster centers while initial centroids are often chosen randomly, and also the chance of selecting one point of each cluster is small. Furthermore the $K-$means algorithm works when the clusters are linearly separable, in addition $K-$means can yield empty clusters.

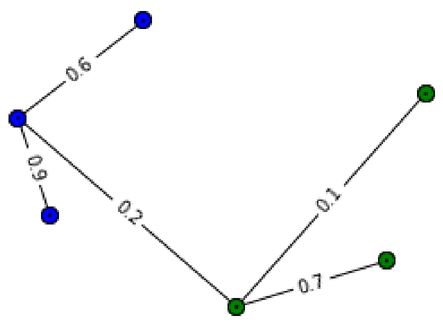
2. Based k-means algorithm in networks

The method of this algorithm is the same as $K-$means except the distance. In fact in data structure for graphs that we consider has no Euclidean distance while the wighted of the shortest path gives the distance. So for recompute the centers we go to each single cluster and find a node which is closer to other nodes inside of the cluster, which basically gives the nodes which has more centrality inside of the cluster.

3. Experimental results



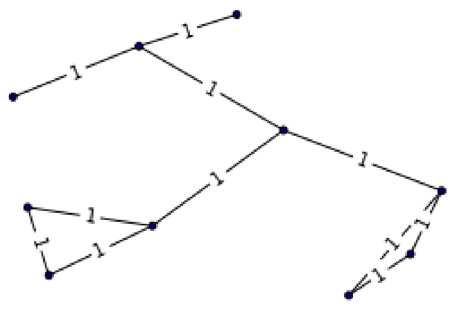
(a) A graph G



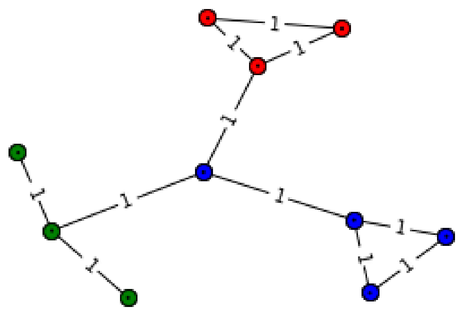
(b)

Figure 1: The out put of algorithm on G for 2 clusters.

G=nx.Graph() G.add_edge(0,1,weight=0.6)
G.add_edge(0,2,weight=0.2)
G.add_edge(2,3,weight=0.1)
G.add_edge(2,4,weight=0.7)
G.add_edge(0,5,weight=0.9)



(a) A graph G



(b)

Figure 2: The out put of algorithm on G for 3 clusters.

```
G=nx.Graph()
G.add_edge(0,1,weight=1)
G.add_edge(0,2,weight=1)
G.add_edge(2,4,weight=1)
G.add_edge(2,5,weight=1)
G.add_edge(0,3,weight=1)
G.add_edge(6,3,weight=1)
G.add_edge(7,3,weight=1)
G.add_edge(8,1,weight=1)
G.add_edge(9,1,weight=1)
G.add_edge(9,8,weight=1)
G.add_edge(6,7,weight=1)
```