CMPT459 Spring 2020
Data Mining
Martin Ester
TAs: Ruijia Mao and Ruchita Rozario

Solution of Assignment 3

Solution posted and discussed in class: March 19, 2020

Assignment 3.1

Consider a 1-dimensional dataset {1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 24, 28, 32, 36, 40} with the three "natural" clusters {1, 2, 3, 4, 5}, {8, 9, 10, 11, 12} and {24, 28, 32, 36, 40}. Apply the k-means algorithm with $k = 3$ to this dataset and show the resulting clusters and cluster centroids for every iteration. Use the algorithm ClusteringByVarianceMinimization presented in the lecture slides, not the incremental algorithm that has originally been published as k-means. We are assuming that ties are broken by assigning an object to the smallest of the closest centroids.

(a) Start with initial centroids of 1, 11, and 28. Does the algorithm detect the natural clusters?

Centroids	1	11	28
Clusters	1, 2, 3, 4, 5	8, 9, 10, 11, 12	24, 28, 32, 36, 40

Centroids	3	10	32
Clusters	1, 2, 3, 4, 5	8, 9, 10, 11, 12	24, 28, 32, 36, 40

Centroids	3	10	32

Yes, k-means detects the natural clusters.

(b) Start with initial centroids of 1, 2, and 3. Does the algorithm detect the natural clusters? What does this tell you?

Centroids	1	2	3
Clusters	1	2	3, 4, 5, 8, 9, 10, 11, 12, 24, 28, 32, 36, 40

Centroids	1	2	17.07
Clusters	1	2, 3, 4, 5, 8, 9	10, 11, 12, 24, 28, 32, 36, 40

Centroids	1	5.17	24.12
Clusters	1, 2, 3	4, 5, 8, 9, 10, 11, 12	24, 28, 32, 36, 40

Centroids	2	8.43	32
Clusters	1, 2, 3, 4, 5	8, 9, 10, 11, 12	24, 28, 32, 36, 40

Centroids	3	10	32
Clusters	1, 2, 3, 4, 5	8, 9, 10, 11, 12	24, 28, 32, 36, 40

Centroids	3	10	32

Yes, k-means again detects the natural clusters. This tells us that even with a bad initialization, k-means may still find the natural clusters, but a poor initialization may increase the number of iterations.

Assignment 3.2

We want to cluster categorical data, i.e. data that have categorical attribute domains. The *k*-medoid algorithm can be applied to any datasets with a given pair-wise distance function and, therefore, is applicable also to categorical data. The *k*-means algorithm, on the other hand, is much more efficient than the *k*-medoid algorithm, but it requires numeric data. The task of this assignment is to develop the equivalent of the *k*-means algorithm for categorical data. We assume the following distance function (Hamming distance) for pairs of categorical objects:

$$dist(x,y) = \sum_{i=1}^{d} \delta(x_i, y_i) \;\; with \;\; \delta(x_i, y_i) = \begin{cases} 0 & if \;\; x_i = y_i \\ 1 & else \end{cases}$$

(a) What is the equivalent of *m* for the means of a cluster *C* for categorical data? Note that *m* must be computable by scanning the set of objects of *C* once (similar to the computation of the cluster means).

As attribute values of *m* we choose the values of the corresponding attribute that occur most frequently in *C*, i.e.

$$m = (m_1, ..., m_d) \text{ such that } \forall\, j, 1 \le j \le d,\; \forall a \in A_j : |\{x \in C \mid x_j = m_j\}| \ge |\{x \in C \mid x_j = a\}|$$

m can be computed in one scan over *C*, since this scan allows us to construct histograms (i.e. frequencies for all values of all attributes) for all *d* attributes.

(b) Prove that *m* according to your definition in (a) is the object minimizing the cluster cost

$$TD(C,m) = \sum_{p \in C} dist(p,m)$$

Hint: first formulate the intuition of the proof, then formalize it. The proof can be performed by contradiction.

Intuitively, this definition of *m* minimizes the cluster cost, since it maximizes the overall number of cases where the attribute value of *m* and the corresponding attribute value of a cluster object *x* agree. Thus, the distance dist(m,x), which counts the number of attributes

More formally, let us assume that there is another object m', $m \neq m'$ with

$$TD(C,m') < TD(C,m).$$

where *x* and *y* disagree, is minimized.

According to our definition of TD and dist,

$$\sum_{i=1}^{d} \sum_{p \in C} \delta(p_i, m'_i) < \sum_{i=1}^{d} \sum_{p \in C} \delta(p_i, m_i)$$

Because of the monotonicity of +, there must exist (at least) one attribute *i* with

$$\sum_{p \in C} \delta(p_i, m'_i) < \sum_{p \in C} \delta(p_i, m_i)$$

i.e., $|\{p \in C \mid p_i = m'_i\}| > |\{p \in C \mid p_i = m_i\}|$

which leads to a contradiction to our definition of *m*.

Assignment 3.3

We want to develop a Divisive (top-down) Hierarchical Clustering method. This method starts with the whole dataset as one cluster. In each iteration, it (1) chooses one of the clusters to split and (2) splits the chosen cluster into two clusters.

(a) How to choose the cluster to be split? Present your solution in plain English and include a rationale for your choice.

There are two major alternatives:
- Split the cluster with the largest number of points. The rationale behind this choice is that it leads to a balanced dendrogram where clusters of one level have approximately the same number of points.
- Split the cluster with the largest variance. The rationale behind this choice is that it leads to clusters with smaller variances. Note that this is, different from partitioning algorithms, not directly the objective of a hierarchical clustering algorithm.

(b) To determine how to split the chosen cluster into two clusters, first consider an optimum split that is the opposite (inverse) of the merge in agglomerative (bottom-up) hierarchical clustering. How is this optimum split defined? Present an (inefficient) algorithm to determine the optimum split. What is the runtime complexity of this algorithm in terms of the number *n* of elements of the cluster to be split?

An optimum split of a set of points C into two subsets C_1 and C_2 is a split that maximizes $dist(S_1, S_2)$.

We enumerate all subsets $S \subseteq C$ and for each of them we determine $D = dist(S, C-S)$. The distance between the two sets of points can be determined using one of the three standard distance functions used in hierarchical clustering, i.e.

$$dist(S, C-S) = \min\{dist(x,y) \mid x \in S, y \in C-S\} \text{ or}$$
$$dist(S, C-S) = \max\{dist(x,y) \mid x \in S, y \in C-S\} \text{ or}$$
$$dist(S, C-S) = \frac{\sum_{x \in S, y \in C-S} dist(x,y)}{|S| \cdot |C-S|}.$$

Finally, we select the split into *S* and *S*-*C* maximizing $D = dist(S, C-S)$.

Since we have to consider all subsets of *C*, the runtime complexity of this algorithm is $O(2^n)$ where $n = |C|$.

(c) Now suggest a more efficient algorithm to find a "good" split approximating the optimum split. What is the runtime complexity of this algorithm in terms of the number *n* of elements of the cluster to be split?

We determine $dist(x,y)$ for all pairs $x \in C, y \in C$. We choose a pair (x', y') with the maximal *dist* value as the seeds of the two subclusters of *C*. Then we assign the remaining points of C to

the closer of the two points and the corresponding subcluster. Since we consider all pairs of points from C, the runtime complexity of this algorithm is $O(n^2)$ where $n = \mid C \mid$.

An even more efficient algorithm splits cluster C by applying k-means with $k = 2$. This algorithm has a runtime complexity of $O(n \cdot i)$ where $n = \mid C \mid$and i = the number of iterations of k-means, and in practise often $O(n \cdot i) = O(n)$.