Clustering - IV

Prof. Dan A. Simovici

UMB

- The k-means algorithm is a partitional algorithm that requires the specification of the number of clusters k as an input.
- The set of objects to be clustered $S = \{\mathbf{o}^1, \ldots, \mathbf{o}^n\}$ is a subset of $\mathbb{R}^m$.

The Starting Point

The k-means algorithm begins with a randomly chosen collection of k points $\mathbf{c}^1, \ldots, \mathbf{c}^k$ in $\mathbb{R}^m$ called centroids. An initial partition of the set S of objects is computed by assigning each object $\mathbf{o}^i$ to its closest centroid $\mathbf{c}^j$.
Let U_j be the set of points assigned to the centroid $\mathbf{c}^j$.
The assignments of objects to centroids are expressed by a matrix (b_{ij}), where

$$b_{ij} = \begin{cases} 1 & \text{if } \mathbf{o}^i \in U_j, \\ 0 & \text{otherwise.} \end{cases}$$

Since each object is assigned to exactly one cluster, we have $\sum_{j=1}^{k} b_{ij} = 1$.
Also, $\sum_{i=1}^{n} b_{ij}$ equals the number of objects assigned to the centroid $\mathbf{c}^j$.

Recomputing the Centroids

After these assignments, expressed by the matrix (b_{ij}), the centroids $\mathbf{c}^j$ must be re-computed using the formula:

$$\mathbf{c}^j = \frac{\sum_{i=1}^{n} b_{ij}\mathbf{o}^i}{\sum_{i=1}^{n} b_{ij}} \tag{1}$$

for $1 \leqslant j \leqslant k$.
The sum of squared errors of a partition $\pi = \{U_1, \ldots, U_k\}$ of a set of objects S was defined as

$$sse(\pi) = \sum_{j=1}^{k} \sum_{\mathbf{o} \in U_j} d^2(\mathbf{o}, \mathbf{c}^j),$$

where $\mathbf{c}^j$ is the centroid of U_j for $1 \leqslant j \leqslant k$. The error of such an assignment is the sum of squared errors of the partition $\pi = \{U_1, \ldots, U_k\}$ defined as

$$sse(\pi) \quad = \quad \sum_{i=1}^{n} \sum_{i=1}^{k} b_{ij} ||\mathbf{o}^i - \mathbf{c}^j||^2$$

The mk necessary conditions for a local minimum of this function,

$$\frac{\partial sse(\pi)}{\partial c_p^j} = \sum_{i=1}^{n} b_{ij} \left(-2(o_p^i - c_p^j)\right) = 0,$$

for $1 \leqslant p \leqslant m$ and $1 \leqslant j \leqslant k$, can be written as

$$\sum_{i=1}^{n} b_{ij} o_p^i = \sum_{i=1}^{n} b_{ij} c_p^j = c_p^j \sum_{i=1}^{n} b_{ij},$$

or as

$$c_p^j = \frac{\sum_{i=1}^{n} b_{ij} o_p^i}{\sum_{i=1}^{n} b_{ij}}$$

for $1 \leqslant p \leqslant m$.

In vectorial form, these conditions amount to

$$\mathbf{c}^j = \frac{\sum_{i=1}^{n} b_{ij}\mathbf{o}^i}{\sum_{i=1}^{n} b_{ij}},$$

which is exactly the formula that is used to update the centroids. Thus, the choice of the centroids can be justified by the goal of obtaining local minima of the sum of squared errors of the clusterings.

Since we have new centroids, objects must be reassigned, which means that the values of b_{ij} must be recomputed, which, in turn, affects the values of the centroids, etc.
The halting criterion of the algorithm depends on particular implementations and may involve

i. performing a certain number of iterations;
ii. lowering the sum of squared errors $sse(\pi)$ below a certain limit;
iii. the current partition coinciding with the previous partition.

Forgy's Algorithm

Algorithm 1: The k-means Forgy's Algorithm

Data: the set of objects to be clustered $S = \{\mathbf{o}^1, \ldots, \mathbf{o}^n\}$ and the number of clusters k

Result: collection of k clusters

1 extract a randomly chosen collection of k vectors $\mathbf{c}_1, \ldots, \mathbf{c}_k$ in $\mathbb{R}^n$;
2 assign each object $\mathbf{o}^i$ to the closest centroid $\mathbf{c}^j$;
3 let $\pi = \{U_1, \ldots, U_k\}$ be the partition defined by $\mathbf{c}^1, \ldots, \mathbf{c}^k$;
4 recompute the centroids of the clusters $U_1, \ldots, U_k$;
5 **while** *halting criterion is not met* **do**
6 compute the new value of the partition π using the current centroids;
7 recompute the centroids of the blocks of π;

Another algorithm, named PAM (an acronym of "Partition Around Medoids") developed by Kaufman and Rousseeuw, also requires as an input parameter the number k of clusters to be extracted.
The k clusters are determined based on a representative object from each cluster, called the *medoid* of the cluster. The medoid of a cluster is one of the objects that have a most central position in the cluster.

PAM begins with a set of objects S, where $|S| = n$, a dissimilarity $n \times n$ matrix D, and a prescribed number of clusters k. The d_{ij} entry of the matrix D is the dissimilarity $d(o_i, o_j)$ between the objects o_i and o_j.

The algorithm has two phases:

- the *building phase*, and
- the swapping phase.

The Builing Phase

The *building phase* aims to construct a set L of *selected objects*, $L \subseteq S$.
The set of *remaining objects* is denoted by R; clearly, $R = S - L$.
To determine the most centrally located object we compute

$$Q_i = \sum_{j=1}^{n} d_{ij}$$

for $1 \leqslant i \leqslant n$.
The most central object o_q is determined by $q = \arg\min_i Q_i$. The set L is initialized as $L = \{o_q\}$.

The Builing Phase (cont'd)

Suppose now that we have constructed a set L of selected objects and $|L| < k$.
To add a new selected object to the set L we need to examine all objects that have not been included in L so far, that is, all objects in R.
The selection is determined by a *merit function* $M : R \longrightarrow \mathbb{N}$.

The Builing Phase (cont'd)

To compute the merit $M(o)$ of an object $o \in R$, we scan all objects in R distinct from o.
Let $o' \in R - \{o\}$ be such an object. If $d(o, o') < d(L, o')$, then adding o to L could benefit the clustering (from the point of view of o') because $d(L, o')$ will diminish.
The potential benefit is $d(o', L) - d(o, o')$. Of course, if $d(o, o') \geqslant d(L, o')$, no such benefit exists (from the point of view of o').
Thus, we compute the merit of o as

$$M(o) = \sum_{o' \in R - \{o\}} \max\{D(L, o') - d(o, o'), 0\}.$$

We add to L the unselected object o that has the largest merit value. The building phase halts when $|L| = k$.
The objects in set L are the potential medoids of the k clusters that we seek to build.

The Swapping Phase

The second phase of the algorithm aims to improve the clustering by considering the merit of swaps between selected and unselected objects.

- In a second phase, swapping objects and existing medoids is considered.
- A cost of a swap is defined with the intention of penalizing swaps that diminish the centrality of the medoids in the clusters.
- Swapping continues as long as useful swaps (that is, swaps with negative costs) can be found.

A cost of a swap is defined with the intention of penalizing swaps that diminish the centrality of the medoids in the clusters. Swapping continues as long as useful swaps (that is, swaps with negative costs) can be found. The second phase of the algorithm aims to improve the clustering by considering the merit of swaps between selected and unselected objects. So, assume now that o_i is a selected object, $o_i \in L$, and o_h is an unselected object, $o_h \in R = S - L$. We need to determine the cost $C(o_i, o_h)$ of swapping o_i and o_h.

Let o_j be an arbitrary unselected object. The contribution c_{ihj} of o_j to the cost of the swap between o_i and o_h is defined as follows:

1. If $d(o_i, o_j)$ and $d(o_h, o_j)$ are greater than $d(o, o_j)$ for any $o \in L - \{o_i\}$, then $c_{ihj} = 0$.
2. If $d(o_i, o_j) = d(L, o_j)$, then two cases must be considered depending on the distance $e(o_j)$ from e_j to the second-closest object of S.
   1. If $d(o_h, o_j) < e(o_j)$, then $c_{ihj} = d(o_h, o_j) - d(S, o_j)$.
   2. If $d(o_h, o_j) \geqslant e(o_j)$, then $c_{ihj} = e(o_j) - d(S, o_j)$.

   In either of these two subcases, we have

$$c_{ihj} = \min\{d(o_h, o_j), e_j\} - d(o_i, o_j).$$

3. If $d(o_i, o_j) > d(L, o_j)$ (that is, o_j is more distant from o_i than from at least one other selected object) and $d(o_h, o_j) < d(L, o_j)$ (which means that o_j is closer to o_h than to any selected object), then $c_{ihj} = d(o_h, o_j) - d(S, o_j)$.

The cost of the swap is $C(o_i, o_h) = \sum_{o_j \in R} c_{ihj}$. The pair that minimizes $C(o_i, o_j)$ is selected. If $C(o_i, o_j) < 0$, then the swap is carried out. All potential swaps are considered.
The algorithm halts when no useful swap exists; that is, no swap with negative cost can be found.

Algorithm 2: The PAM algorithms

Data: a set of objects S, where $|S| = n$, a dissimilarity $n \times n$ matrix D, and a prescribed number of clusters k

Result: a k-clustering of S

1 construct the set L of k medoids;
2 **repeat**
3 | compute the costs $C(o_i, o_h)$ for $o_i \in L$ and $o_h \in R$;
4 | select the pair (o_i, o_h) that corresponds to the minimum $m = C(o_i, o_h)$;
5 **until** $(m > 0)$;

Note that inside the loop **repeat** $\cdots$ **until** there are $l(n-l)$ pairs of objects to be examined, and for each pair we need to involve $n-l$ non-selected objects. Thus, one execution of the loop requires $O(l(n-l)^2)$, and the total execution may require up to $O\left(\sum_{l=1}^{n-l} l(n-l)^2\right)$, which is $O(n^4)$. Thus, the usefulness of PAM is limited to rather small data set (no more than a few hundred objects).