1 Cluster Analysis (unsupervised learning)

1.1 Metric clustering, the k-means method

Here we consider a method of finding groups (clusters) of data points in a finite dimensional Euclidean space. Given the points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and an integer $1 < k < n$, we are looking for the k-partition of the index set $\{1, \ldots, n\}$ (or equivalently, the clustering of the points into k disjoint non-empty subsets) which minimizes the following *k-variance* of the points over all possible k-partitions $P_k = (C_1, \ldots, C_k)$:

$$S_k^2(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \min_{P_k} S_k^2(P_k; \mathbf{x}_1, \ldots, \mathbf{x}_n) = \min_{P_k} \sum_{a=1}^{k} \sum_{j \in C_a} \|\mathbf{x}_j - \mathbf{c}_a\|^2 \qquad (1)$$

where $\mathbf{c}_a = \frac{1}{|C_a|} \sum_{j \in C_a} \mathbf{x}_j$ is the center of cluster a ($a = 1, \ldots, k$).

In general, $d \leq k$, and they are much less than n. In fact, the above k-variance corresponds to the sum of the within-cluster variances, that is to ($\mathrm{tr}\mathbf{W}$) in the MANOVA decomposition

$$\mathbf{T} = \mathbf{W} + \mathbf{B}.$$

Since the total variance ($\mathrm{tr}\mathbf{T}$) is fixed, minimizing

$$\frac{\mathrm{tr}(\mathbf{W})}{\mathrm{tr}(\mathbf{B})} = \frac{\mathrm{tr}(\mathbf{W})}{\mathrm{tr}(\mathbf{T}) - \mathrm{tr}(\mathbf{W})}$$

is equivalent to minimizing $\mathrm{tr}(\mathbf{W})$, i.e. S_k^2.

To find the global minimum is NP-complete, but the iteration of the k-means algorithm is capable to find a local minimum in polynomial time. The vectors $\mathbf{c}_1, \ldots, \mathbf{c}_k$ are usually referred to as the *centroids* of the clusters, and in a more abstract formulation of the above optimization task, e.g., in the work of MacQueen, they are also looked for. Roughly speaking, starting with an initial clustering, the iteration of the simple k-means algorithm consists of the following two alternating steps.

- In the first step, fixing the clustering of the points, it finds the cluster centers (they will be the mass centers by the Steiner's theorem).
- In the second step, the algorithm relocates the points in such a way that it assigns a point to the cluster, the center of which is the closest to it (in case of ambiguity the algorithm chooses the smallest index such cluster).

If there exists a well-separated k-clustering of the points (even the largest intra-cluster distance is smaller than the smallest inter-cluster one) the convergence of the algorithm to the global minimum can be proved, with a convenient starting.

Sometimes the points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are endowed with the positive weights $d_1, \ldots, d_n$, where without loss of generality $\sum_{i=1}^{n} d_i = 1$ can be assumed. In such cases the *weighted k-variance* of the points

$$\tilde{S}_k^2(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \min_{P_k} \tilde{S}_k^2(P_k; \mathbf{x}_1, \ldots, \mathbf{x}_n) = \min_{P_k} \sum_{a=1}^{k} \sum_{j \in C_a} d_j \|\mathbf{x}_j - \mathbf{c}_a\|^2 \qquad (2)$$

is considered, where $\mathbf{c}_a = \frac{1}{\sum_{j \in C_a} d_j} \sum_{j \in C_a} d_j \mathbf{x}_j$ is the weighted center of cluster a ($a = 1, \ldots, k$). The above algorithm can be easily adapted to this situation. Note that $\tilde{S}_k^2(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ corresponds to the k-variance with respect to the distribution $d_1, \ldots, d_n$. In this contexts, $S_k^2(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is the special case when this law is uniform.

Likewise, instead of L^2-distances, other kind of distance functions in the objective function can be used. Sometimes coordinate-wise medians are used instead of mass centers (k-medoid algorithm).

A well-known drawback of the k-means clustering is that the clusters need to be convex in order to achieve satisfactory results. That is, the k-means algorithm forms spherical clusters whether or not the underlying data distribution obeys this form. Otherwise, our data can be mapped into a feature space and we apply k-means clustering for the mapped data which already have this spherical structure. When we use these so-called Reproducing Kernel Hilbert Spaces, we need not actually map our data, but can find the squared Euclidean distance between a feature point $\phi(\mathbf{x}_\ell)$ and the center $\mathbf{c}$ of its cluster C via the kernel K in the following way:

$$\begin{aligned}
&\|\phi(\mathbf{x}_\ell) - \mathbf{c}\|^2 = \\
&= \langle \phi(\mathbf{x}_\ell), \phi(\mathbf{x}_\ell) \rangle + \frac{1}{|C|^2} \sum_{i \in C} \sum_{j \in C} \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle - \frac{2}{|C|} \sum_{i \in C} \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_\ell) \rangle \\
&= K(\mathbf{x}_\ell, \mathbf{x}_\ell) + \frac{1}{|C|^2} \sum_{i \in C} \sum_{j \in C} K(\mathbf{x}_i, \mathbf{x}_j) - \frac{2}{|C|} \sum_{i \in C} K(\mathbf{x}_i, \mathbf{x}_\ell).
\end{aligned}$$

1.2 Hierarchical clustering

Here we start with pairwise distances or similarities of the points, which are not necessarily in a metric space. For the definition of a distance matrix see Definition 1, whereas pairwise similarities can be thought of as absolute values of correlations.

We distinguish between *agglomerative* or *divisive* methods, depending on, whether we start with singleton clusters (each point forms a cluster) and we aggregate them step by step, or we start with one cluster (containing all of the points) and we separate them step by step.

Consider the agglomerative method. Based on the pairwise distances d_{ij}'s, whatever the clusters are, we always aggregate the two closest ones into one cluster. The distance of the (disjoint) clusters T and H can be defined in the following ways:

- *Single linkage (nearest neighbor)*:
$$d(C, H) = \min_{i \in C,\, j \in H} d_{ij}.$$
- *Complete linkage (farthest neighbor)*:
$$d(C, H) = \max_{i \in C,\, j \in H} d_{ij}.$$
- *Average linkage*:
$$d(C, H) = \frac{1}{|C| \cdot |H|} \sum_{i \in C} \sum_{j \in H} d_{ij}.$$

Based on the agglomeration, a *dendrogram* is made, based on which the user selects the best clustering.

2 Multidimensional Scaling (MDS)

For clustering purposes, sometimes we have abstract objects which are basically not in a metric space, and all we have, are their pairwise distances. In a broad sense, the notion of a *distance matrix* in the upcoming definition allows to define distances between objects subjectively, possibly as monotonous decreasing functions of their pairwise similarities.

Definition 1 *The $n \times n$ matrix $\mathbf{D}$ is called distance matrix if it is symmetric and*

$$d_{ii} = 0 \quad (i = 1, \ldots, n), \quad \text{and} \quad d_{ij} = d_{ji} \geq 0 \quad (i \neq j).$$

This definition allows the entries of the distance matrix not to obey the triangle inequalities, and even if they do so, the metric defined by the distances is not necessarily the Euclidean one.

Our purpose is to find a dimension d and points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ such that their pairwise Euclidean distances approach the entries of the distance matrix as much as possible. Of course, if the distances are real distances of objects in a physical space, the method of Multidimensional Scaling to be introduced is assumed to find the dimension and configuration of the hidden points, at least, up to translation, rotation, and reflection. For example, if someone provides us with the pairwise Euclidean distances of cities (not too far apart), we will be able to reconstruct their mutual position. Even in this case, our measurements may be subject to error, therefore, we want to find a solution to the problem, which is able to give a good approximation in any of the above cases.

The situation when the distances can exactly be realized in a Euclidean space is defined now.

Definition 2 *The $n \times n$ distance matrix is Euclidean if there is a positive integer d and points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ such that*

$$\|\mathbf{x}_i - \mathbf{x}_j\| = d_{ij}, \quad i, j = 1, \ldots, n.$$

The following theorem gives a necessary and sufficient condition for a distance matrix to be Euclidean. Intuitively, one needs somehow to eliminate the translation invariance, therefore makes use of the so-called *centering matrix* $\mathbf{C}_n = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$ (we will drop the index n in the sequel).

Theorem 1 *Given an $n \times n$ distance matrix $\mathbf{D}$, the matrix $\mathbf{A} = (a_{ij})$ is defined as $a_{ij} = -\frac{1}{2}d_{ij}^2$ $(i, j = 1, \ldots, n)$. The matrix $\mathbf{D}$ is Euclidean if and only if the symmetric matrix $\mathbf{CAC}$ is positive semidefinite.*

We just remark that the construction for the point configuration, under the conditions of Theorem 1 , is the following. The dimension will be the rank of $\mathbf{A}$, i.e. $d = \operatorname{rank}(\mathbf{A})$. Let $\lambda_1 \geq \cdots \geq \lambda_d > 0$ be the strictly positive eigenvalues of $\mathbf{A}$ with corresponding unit-norm eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_d$. The points will be row vectors of the $n \times d$ matrix $(\sqrt{\lambda_1}\mathbf{u}_1, \ldots, \sqrt{\lambda_d}\mathbf{u}_d)$. In fact, this idea is inspired by the theory of Gram-matrices. This system is unique apart

from translation, rotation, and reflection, not to mention the indeterminacy due to possible multiple eigenvalues and the triviality that the coordinates can be inflated with any number of zero coordinates.

Note that the construction also gives a hint how to find an approximate solution when $\mathbf{D}$ is not Euclidean, but not 'far' from that, i.e. the matrix $\mathbf{CAC}$ has some slightly negative eigenvalues. Then we omit those and use the eigenvectors corresponding to the positive ones in the above construction. Even if $\mathbf{D}$ is Euclidean, the rank of $\mathbf{CAC}$ may be so large that we want a smaller dimensional configuration that reconstructs the distances with a tolerable error. For this purpose, we retain the largest eigenvalues is the analysis, better to stay, we look for a gap in the spectrum so that the eigenvalues behind this gap are negligible compared to those before the gap. Then the number of the outstanding eigenvalues will be the dimension of the points we look for.

Note that the Spectral Clustering looks for clusters of the vertices of a graph or hypergraph based on the spectral decomposition of their adjacency, weighted adjacency, Laplacian, or modularity matrices.