Clustering

SC4/SM4 Data Mining and Machine Learning, Hilary Term 2017

Dino Sejdinovic

Clustering is one of the fundamental and ubiquitous tasks in exploratory data analysis – a first intuition about the data is often based on identifying meaningful disjoint groups among the data items. In *partition-based* clustering, which we consider in this note, one divides n data items into K clusters $C_1, \ldots, C_K$ where for all $k, k' \in \{1, \ldots, K\}$,

$$C_k \subset \{1, \ldots, n\}, \qquad C_k \cap C_{k'} = \emptyset \ \ \forall k \neq k', \qquad \bigcup_{k=1}^{K} C_k = \{1, \ldots, n\}.$$

Central to the goals of clustering is the notion of similarity/dissimilarity between data items. There will be many ways to define the notion of similarity, and the choice will depend on the dataset being analyzed and dictated by domain specific knowledge.

Intuitively, clustering aims to group similar items together and to place separate dissimilar items into different groups. However, note that these two objectives in many cases contradict each other (similarity is not a transitive relation, while being in the same cluster is an equivalence relation). One could imagine a long sequence of items such that each next item is very similar to the previous one so that they should all belong to the same cluster – but that would also mean that the endpoints are potentially highly dissimilar. Hence, there are also different clustering techniques which emphasize different aspects of these goals, i.e. whether to keep similar points together or dissimilar points apart.

There have been several attempts to construct an axiomatic definition of clustering, but it is surprisingly difficult to put on rigorous footing. Consider the following three basic properties required of a clustering method $\mathcal{F} : (\mathcal{D} = \{x_i\}_{i=1}^n, \rho) \mapsto \{C_1, \ldots, C_K\}$ which takes as an input dataset $\mathcal{D}$ and a dissimilarity function ρ and returns a partition of $\mathcal{D}$:

- **Scale invariance.** For any $\alpha > 0$, $\mathcal{F}(\mathcal{D}, \alpha\rho) = \mathcal{F}(\mathcal{D}, \rho)$, i.e. partition should not depend on units in which dissimilarity is measured.
- **Richness.** For any partition $C = \{C_1, \ldots, C_K\}$ of $\mathcal{D}$, there exists dissimilarity ρ, such that $\mathcal{F}(\mathcal{D}, \rho) = C$, i.e. the outcome is fully controlled by the dissimilarity function.
- **Consistency.** If ρ and ρ' are two dissimilarities such that for all $x_i, x_j \in \mathcal{D}$ the following holds:

  $$\begin{aligned} x_i, x_j \text{ belong to the same cluster in } \mathcal{F}(\mathcal{D}, \rho) &\implies \rho'(x_i, x_j) \leq \rho(x_i, x_j) \\ x_i, x_j \text{ belong to different clusters in } \mathcal{F}(\mathcal{D}, \rho) &\implies \rho'(x_i, x_j) \geq \rho(x_i, x_j), \end{aligned}$$

  then $\mathcal{F}(\mathcal{D}, \rho') = \mathcal{F}(\mathcal{D}, \rho)$. In other words, if the items in the same cluster become more similar and the items already separated become less similar, then the clustering should not change.

While all three properties appear natural, Kleinberg's impossibility theorem [1] states that there exists no clustering method that satisfies all three properties, implying that every clustering method will have some undesirable properties. For further discussion, see Section 22.5 in [4].

We will consider three widely used clustering methods: K-means algorithm (and its extension, DP-means), spectral clustering and hierarchical clustering.

2.1 K-means algorithm

K-means is the simplest partition-based clustering algorithm. It uses a preassigned number of clusters and represents each cluster using a *prototype* or *cluster centroid* μ_k.

The idea of K-means is to measure the quality of each cluster using its *within-cluster deviance* from the cluster centroids

$$W(C_k, \mu_k) = \sum_{i \in C_k} \|x_i - \mu_k\|_2^2.$$

The overall quality of the clustering is then given by the total within-cluster deviance:

$$W(\{C_k\}, \{\mu_k\}) = \sum_{k=1}^{K} W(C_k, \mu_k) = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 = \sum_{i=1}^{n} \|x_i - \mu_{c_i}\|_2^2,$$

where $c_i = k$ if and only if $i \in C_k$. This is now the overall objective function used to select both the cluster centroids and the assignment of points to clusters. The joint optimization over both the partition $\{C_k\}$ and centroids $\{\mu_k\}$ is a combinatorial optimization problem and is computationally hard. However, note that

- Given partition $\{C_k\}$, we can easily find the optimal centroids by differentiating W with respect to μ_k:

$$\frac{\partial W}{\partial \mu_k} = 2 \sum_{i \in C_k} (x_i - \mu_k) = 0 \qquad \Rightarrow \mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

- Given prototypes, we can easily find the optimal partition by assigning each data point to the closest cluster prototype:

$$c_i = \operatorname*{argmin}_k \|x_i - \mu_k\|_2^2 .$$

Thus one can employ an iterative alternating optimization, which is exactly the K-means algorithm:

K-means algorithm.

1. Randomly initialize K cluster centroids $\mu_1, \dots, \mu_K$.
2. *Cluster assignment:* For each $i = 1, \dots, n$, assign each x_i to the cluster with the nearest centroid,

$$c_i := \operatorname*{argmin}_{k=1,\dots,K} \|x_i - \mu_k\|_2^2$$

   Set $C_k := \{i : c_i = k\}$ for each k.

3. *Move centroids:* Set $\mu_1, \ldots, \mu_K$ to the averages of the new clusters:

$$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

4. Repeat steps 2-3 until convergence.

5. Return the partition $\{C_1, \ldots, C_K\}$ and means $\mu_1, \ldots, \mu_K$.

K-means is a heuristic search algorithm so it can (and often will) get stuck at local optima. The result depends on the starting configurations. Typically one performs a number of runs from different random initial values of centroids, and then chooses the end result with minimum W. Since each step does not increase the objective function and the number of possible partitions is finite, the algorithm will converge to a local optimum. However, note that there could be ties in the cluster assignment, which need to be broken in a systematic fashion.

2.2 DP-means

K-means is intuitive and straightforward to implement, but how do we select the number of clusters K in the first place? Clearly, the objective function is minimized (and equals zero) if we let $K = n$, but this is not a meaningful clustering.

One elegant approach is the *DP-means algorithm* [2] that comes from the interpretation of K-means using small variance asymptotics of the Expectation Maximization (EM) algorithm for mixture modelling. We will discuss mixture modelling and EM algorithm later in the course. DP-means starts from a single cluster, i.e. $K = 1$ and modifies the cluster assignment step as follows:

1. Initialize $K = 1$ and $\mu_1 = \frac{1}{n} \sum_{i=1}^{n} x_i$ (the global mean).
2. *DP-means cluster assignment:* For each $i = 1, \ldots, n$,
   - if $\min_{k=1,\ldots,K} \|x_i - \mu_k\|_2^2 > \lambda$, set $K \leftarrow K + 1$, $c_i \leftarrow K$, $\mu_K \leftarrow x_i$
   - otherwise, set $c_i = \operatorname{argmin}_{k=1,\ldots,K} \|x_i - \mu_k\|_2^2$.

The rest of the algorithm is exactly the same as K-means. Tuning parameter λ controls the tradeoff between the traditional K-means objective and the number of clusters. DP-means can be shown to locally minimize the objective

$$W_\lambda(\{C_k\}, \{\mu_k\}, K) = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 + \lambda K. \tag{2.1}$$

Indeed, just like in K-means algorithm, the "move centroids" step can only decrease the objective, whereas for every data item $i = 1, \ldots, n$, its assignment to the nearest centroid if closer than λ will not increase the objective and if the nearest centroid is at a distance larger than λ we can create another cluster and pay a penalty λ while still decreasing the overall objective (2.1).

2.3 Spectral Clustering

2.3.1 Clustering and Graph Cuts

K-means algorithm will often fail when applied to data with elongated or non-convex cluster structures. An alternative approach to clustering is to use *graph cuts* on a weighted undirected *similarity* graph $G = (\{1, \ldots, n\}, \mathbf{W})$ induced by the dataset consisting of n observations $\{x_i\}_{i=1}^n$. The n vertices represent the observations and pairs of vertices are connected by an edge if their similarity exceeds some threshold. This is based on a similarity (affinity/kernel) matrix $\mathbf{W}$ (e.g., a non-negative kernel $k(x_i, x_j)$ can be used as the graph weight w_{ij}, or one could consider nearest neighbour graphs leading to *sparse* similarity matrices). We wish to partition the dataset into K clusters, which can be thought of as a partition $C_1, C_2, \ldots, C_K$ of the vertex set $\{1, \ldots, n\}$. The overall *graph cut* across clusters is given by

$$\operatorname{cut}(C_1, \ldots, C_K) = \sum_{k=1}^{K} \operatorname{cut}(C_k, \bar{C}_k),$$

where $\bar{C}_k$ is the complement of C_k and $\operatorname{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij}$ is the sum of the weights separating vertex subset A from the vertex subset B, where A and B are disjoint.

The result of cut minimization, however, results with very small cluster sizes (i.e. it would typically split a single datapoint from the rest), so one needs to balance the cuts by the cluster sizes in the partition. One approach is to consider the notion of "ratio cut"

$$\text{ratio-cut}(C_1, \ldots, C_K) = \sum_{k=1}^{K} \frac{\operatorname{cut}(C_k, \bar{C}_k)}{|C_k|}.$$

Unfortunately, minimizing this criterion is computationally hard to solve. Spectral clustering algorithm uses a relaxation of the problem of minimizing the ratio cut.

2.3.2 Graph Laplacian

Definition 2.1. The *(unnormalized) Laplacian* of a graph $G = (\{1, \ldots, n\}, \mathbf{W})$ is an $n \times n$ matrix given by

$$\mathbf{L} = \mathbf{D} - \mathbf{W},$$

where $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{ii} = \deg(i)$, and $\deg(i)$ denotes the *degree* of vertex i defined as

$$\deg(i) = \sum_{j=1}^{n} w_{ij}.$$

Note that the Laplacian always has the column vector $\mathbf{1}$ as an eigenvector with eigenvalue 0 (since all rows sum to zero).

Exercise 2.1. For all $a \in \mathbb{R}^n$

$$a^\top \mathbf{L} a = \frac{1}{2} \sum_{i,j} w_{ij} (a_i - a_j)^2 \geq 0,$$

which means that the Laplacian is a positive semi-definite matrix, and all the eigenvalues are non-negative.

2.3.3 Laplacian and Ratio Cuts

The relationship between the ratio cuts and the graph Laplacian is given in the following:

Lemma 2.1. For a given partition $C_1, C_2, \ldots, C_K$ define the column vectors $h_k \in \mathbb{R}^n$ as

$$h_{k,i} = \frac{1}{\sqrt{|C_k|}} \mathbf{1}_{\{i \in C_k\}}.$$

Then

$$\text{ratio-cut}\,(C_1, \ldots, C_K) = \sum_{k=1}^{K} h_k^\top \mathbf{L} h_k. \tag{2.2}$$

Note that the vectors h_k are orthonormal by construction. Thus, to minimize the ratio cut exactly, we can search for orthonormal vectors h_k with entries either 0 or $1/\sqrt{|C_k|}$ which minimize the RHS in (2.2). This is equivalent to integer programming so it is computationally hard. Thus, we instead look for *any collection of orthonormal vectors* h_k that minimize RHS in (2.2) – which corresponds to the eigendecomposition of the Laplacian.

If the original graph is disconnected, in addition to $\mathbf{1}$, there would be other 0-eigenvectors of $\mathbf{L}$, corresponding to the indicators of the connected components of the graph (see Theorem 25.4.1 in [3]). The idea of spectral clustering is to assume that the graph we end up with based on the dataset, while possibly not disconnected, is a “small perturbation” of a disconnected graph, and we are trying to recover connected components, i.e., clusters based on a noisy version of the true Laplacian of the underlying disconnected graph.

The algorithm proceeds by eigendecomposing $\mathbf{L}$ and taking the K eigenvectors corresponding to the K smallest eigenvalues – this gives a new ”data representation”

$$\mathbf{Z} = [u_1, \ldots, u_K] \in \mathbb{R}^{n \times K}$$

on which we can apply a more conventional clustering algorithm, such as K-means.

Remark 2.1. A number of *normalized* graph Laplacians have also been proposed, which are based on slightly different “balancing” formulation of cuts, including the “random walk” matrix $\mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$ and $\mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$.

[5] provides an in-depth overview of spectral clustering.

2.4 Hierarchical Clustering

Hierarchical clustering is an iterative procedure that constructs a hierarchy of clusters, starting with individual data points and merging them into clusters when their dissimilarity is below a given threshold and then merging lower level clusters into higher level clusters. This process can be visualised by a tree/dendrogram. Dendrograms depict cluster assignments with respect to increasing values of dissimilarity threshold. Cutting a dendrogram horizontally at a particular height partitions the data into disjoint clusters which are represented by the vertical lines it intersects.

To join clusters C_i and C_j into a higher level cluster, we need a way to measure the dissimilarity $D(C_i, C_j)$ between them based on the dissimilarity d between the individual data items. There are several approaches:

(a) *Single Linkage*: elongated, loosely connected clusters

$$D(C_i, C_j) = \min_{x,y} \left(d(x,y) | x \in C_i, y \in C_j \right),$$

(b) *Complete Linkage*: compact clusters, relatively similar objects can remain separated at high levels

$$D(C_i, C_j) = \max_{x,y} \left(d(x,y) | x \in C_i, y \in C_j \right),$$

(c) *Average Linkage*: tries to balance the two above, but affected by the scale of dissimilarities

$$D(C_i, C_j) = \text{avg}_{x,y} \left(d(x,y) | x \in C_i, y \in C_j \right).$$

References

[1] Jon M. Kleinberg. An impossibility theorem for clustering. In *Advances in Neural Information Processing Systems 15*, pages 463–470. MIT Press, 2003.

[2] Brian Kulis and Michael I. Jordan. Revisiting k-means: New Algorithms via Bayesian Nonparametrics. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 513–520, 2012.

[3] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012.

[4] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, New York, NY, USA, 2014.

[5] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.