22

Clustering

Clustering is one of the most widely used techniques for exploratory data analysis. Across all disciplines, from social sciences to biology to computer science, people try to get a first intuition about their data by identifying meaningful groups among the data points. For example, computational biologists cluster genes on the basis of similarities in their expression in different experiments; retailers cluster customers, on the basis of their customer profiles, for the purpose of targeted marketing; and astronomers cluster stars on the basis of their spacial proximity.

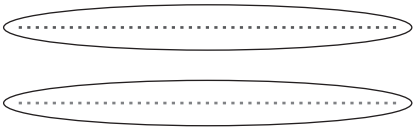
The first point that one should clarify is, naturally, what is clustering? Intuitively, clustering is the task of grouping a set of objects such that similar objects end up in the same group and dissimilar objects are separated into different groups. Clearly, this description is quite imprecise and possibly ambiguous. Quite surprisingly, it is not at all clear how to come up with a more rigorous definition.

There are several sources for this difficulty. One basic problem is that the two objectives mentioned in the earlier statement may in many cases contradict each other. Mathematically speaking, similarity (or proximity) is not a transitive relation, while cluster sharing is an equivalence relation and, in particular, it is a transitive relation. More concretely, it may be the case that there is a long sequence of objects, $x_1, \ldots, x_m$ such that each x_i is very similar to its two neighbors, x_{i-1} and x_{i+1}, but x_1 and x_m are very dissimilar. If we wish to make sure that whenever two elements are similar they share the same cluster, then we must put all of the elements of the sequence in the same cluster. However, in that case, we end up with dissimilar elements (x_1 and x_m) sharing a cluster, thus violating the second requirement.

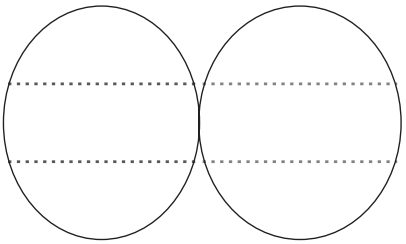
To illustrate this point further, suppose that we would like to cluster the points in the following picture into two clusters.

A clustering algorithm that emphasizes not separating close-by points (e.g., the Single Linkage algorithm that will be described in Section 22.1) will cluster this input

by separating it horizontally according to the two lines:

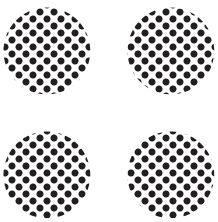


In contrast, a clustering method that emphasizes not having far-away points share the same cluster (e.g., the 2-means algorithm that will be described in Section 22.1) will cluster the same input by dividing it vertically into the right-hand half and the left-hand half:

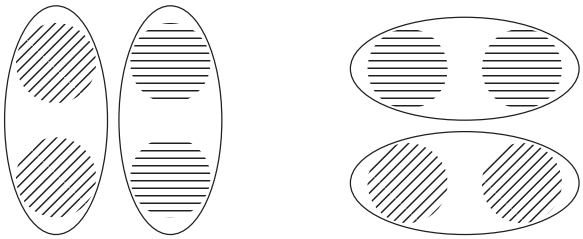


Another basic problem is the lack of "ground truth" for clustering, which is a common problem in *unsupervised learning*. So far in the book, we have mainly dealt with *supervised* learning (e.g., the problem of learning a classifier from labeled training data). The goal of supervised learning is clear – we wish to learn a classifier which will predict the labels of future examples as accurately as possible. Furthermore, a supervised learner can estimate the success, or the risk, of its hypotheses using the labeled training data by computing the empirical loss. In contrast, clustering is an *unsupervised learning* problem; namely, there are no labels that we try to predict. Instead, we wish to organize the data in some meaningful way. As a result, there is no clear success evaluation procedure for clustering. In fact, even on the basis of full knowledge of the underlying data distribution, it is not clear what is the "correct" clustering for that data or how to evaluate a proposed clustering.

Consider, for example, the following set of points in $\mathbb{R}^2$:



and suppose we are required to cluster them into two clusters. We have two highly justifiable solutions:



This phenomenon is not just artificial but occurs in real applications. A given set of objects can be clustered in various different meaningful ways. This may be due to having different implicit notions of distance (or similarity) between objects, for example, clustering recordings of speech by the accent of the speaker versus clustering them by content, clustering movie reviews by movie topic versus clustering them by the review sentiment, clustering paintings by topic versus clustering them by style, and so on.

To summarize, there may be several very different conceivable clustering solutions for a given data set. As a result, there is a wide variety of clustering algorithms that, on some input data, will output very different clusterings.

A Clustering Model:

Clustering tasks can vary in terms of both the type of input they have and the type of outcome they are expected to compute. For concreteness, we shall focus on the following common setup:

Input – a set of elements, $\mathcal{X}$, and a distance function over it. That is, a function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ that is symmetric, satisfies $d(x,x) = 0$ for all $x \in \mathcal{X}$, and often also satisfies the triangle inequality. Alternatively, the function could be a similarity function $s : \mathcal{X} \times \mathcal{X} \to [0,1]$ that is symmetric and satisfies $s(x,x) = 1$ for all $x \in \mathcal{X}$. Additionally, some clustering algorithms also require an input parameter k (determining the number of required clusters).

Output – a partition of the domain set $\mathcal{X}$ into subsets. That is, $C = (C_1, \ldots C_k)$ where $\bigcup_{i=1}^{k} C_i = \mathcal{X}$ and for all $i \neq j$, $C_i \cap C_j = \emptyset$. In some situations the clustering is "soft," namely, the partition of $\mathcal{X}$ into the different clusters is probabilistic where the output is a function assigning to each domain point, $x \in \mathcal{X}$, a vector $(p_1(x), \ldots, p_k(x))$, where $p_i(x) = \mathbb{P}[\mathbf{x} \in C_i]$ is the probability that $\mathbf{x}$ belongs to cluster C_i. Another possible output is a clustering *dendrogram* (from Greek dendron = tree, gramma = drawing), which is a hierarchical tree of domain subsets, having the singleton sets in its leaves, and the full domain as its root. We shall discuss this formulation in more detail in the following.

In the following we survey some of the most popular clustering methods. In the last section of this chapter we return to the high level discussion of what is clustering.

22.1 LINKAGE-BASED CLUSTERING ALGORITHMS

Linkage-based clustering is probably the simplest and most straightforward paradigm of clustering. These algorithms proceed in a sequence of rounds. They

start from the trivial clustering that has each data point as a single-point cluster. Then, repeatedly, these algorithms merge the "closest" clusters of the previous clustering. Consequently, the number of clusters decreases with each such round. If kept going, such algorithms would eventually result in the trivial clustering in which all of the domain points share one large cluster. Two parameters, then, need to be determined to define such an algorithm clearly. First, we have to decide how to measure (or define) the distance between clusters, and, second, we have to determine when to stop merging. Recall that the input to a clustering algorithm is a between-points distance function, d. There are many ways of extending d to a measure of distance between domain subsets (or clusters). The most common ways are

1. Single Linkage clustering, in which the between-clusters distance is defined by the minimum distance between members of the two clusters, namely,

$$D(A, B) \stackrel{\text{def}}{=} \min\{d(x, y) : x \in A,\ y \in B\}$$

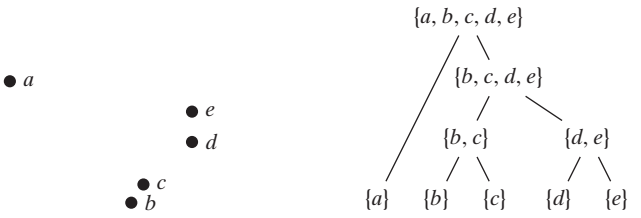
2. Average Linkage clustering, in which the distance between two clusters is defined to be the average distance between a point in one of the clusters and a point in the other, namely,

$$D(A, B) \stackrel{\text{def}}{=} \frac{1}{|A||B|} \sum_{x \in A,\ y \in B} d(x, y)$$

3. Max Linkage clustering, in which the distance between two clusters is defined as the maximum distance between their elements, namely,

$$D(A, B) \stackrel{\text{def}}{=} \max\{d(x, y) : x \in A,\ y \in B\}.$$

The linkage-based clustering algorithms are *agglomerative* in the sense that they start from data that is completely fragmented and keep building larger and larger clusters as they proceed. Without employing a stopping rule, the outcome of such an algorithm can be described by a clustering *dendrogram*: that is, a tree of domain subsets, having the singleton sets in its leaves, and the full domain as its root. For example, if the input is the elements $\mathcal{X} = \{a, b, c, d, e\} \subset \mathbb{R}^2$ with the Euclidean distance as depicted on the left, then the resulting dendrogram is the one depicted on the right:



The single linkage algorithm is closely related to Kruskal's algorithm for finding a minimal spanning tree on a weighted graph. Indeed, consider the full graph whose vertices are elements of $\mathcal{X}$ and the weight of an edge (x, y) is the distance $d(x, y)$. Each merge of two clusters performed by the single linkage algorithm corresponds to a choice of an edge in the aforementioned graph. It is also possible to show that

the set of edges the single linkage algorithm chooses along its run forms a minimal spanning tree.

If one wishes to turn a dendrogram into a partition of the space (a clustering), one needs to employ a *stopping criterion*. Common stopping criteria include

- Fixed number of clusters – fix some parameter, k, and stop merging clusters as soon as the number of clusters is k.
- Distance upper bound – fix some $r \in \mathbb{R}_+$. Stop merging as soon as all the between-clusters distances are larger than r. We can also set r to be $\alpha \max\{d(x,y) : x, y \in \mathcal{X}\}$ for some $\alpha < 1$. In that case the stopping criterion is called "scaled distance upper bound."

22.2 k-MEANS AND OTHER COST MINIMIZATION CLUSTERINGS

Another popular approach to clustering starts by defining a cost function over a parameterized set of possible clusterings and the goal of the clustering algorithm is to find a partitioning (clustering) of minimal cost. Under this paradigm, the clustering task is turned into an optimization problem. The objective function is a function from pairs of an input, $(\mathcal{X}, d)$, and a proposed clustering solution $C = (C_1, \ldots, C_k)$, to positive real numbers. Given such an objective function, which we denote by G, the *goal* of a clustering algorithm is defined as finding, for a given input $(\mathcal{X}, d)$, a clustering C so that $G((\mathcal{X}, d), C)$ is minimized. In order to reach that goal, one has to apply some appropriate search algorithm.

As it turns out, most of the resulting optimization problems are NP-hard, and some are even NP-hard to approximate. Consequently, when people talk about, say, k-means clustering, they often refer to some particular common approximation algorithm rather than the cost function or the corresponding exact solution of the minimization problem.

Many common objective functions require the number of clusters, k, as a parameter. In practice, it is often up to the user of the clustering algorithm to choose the parameter k that is most suitable for the given clustering problem.

In the following we describe some of the most common objective functions.

The k-means objective function is one of the most popular clustering objectives. In k-means the data is partitioned into disjoint sets $C_1, \ldots, C_k$ where each C_i is represented by a centroid μ_i. It is assumed that the input set $\mathcal{X}$ is embedded in some larger metric space $(\mathcal{X}', d)$ (so that $\mathcal{X} \subseteq \mathcal{X}'$) and centroids are members of $\mathcal{X}'$. The k-means objective function measures the squared distance between each point in $\mathcal{X}$ to the centroid of its cluster. The centroid of C_i is defined to be

$$\mu_i(C_i) = \operatorname*{argmin}_{\mu \in \mathcal{X}'} \sum_{x \in C_i} d(x, \mu)^2.$$

Then, the k-means objective is

$$G_{\text{k-means}}((\mathcal{X}, d), (C_1, \ldots, C_k)) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i(C_i))^2.$$

This can also be rewritten as

$$G_{\text{k-means}}((\mathcal{X},d),(C_1,\ldots,C_k)) = \min_{\mu_1,\ldots\mu_k\in\mathcal{X}'} \sum_{i=1}^{k}\sum_{x\in C_i} d(x,\mu_i)^2. \tag{22.1}$$

The k-means objective function is relevant, for example, in digital communication tasks, where the members of $\mathcal{X}$ may be viewed as a collection of signals that have to be transmitted. While $\mathcal{X}$ may be a very large set of real valued vectors, digital transmission allows transmitting of only a finite number of bits for each signal. One way to achieve good transmission under such constraints is to represent each member of $\mathcal{X}$ by a "close" member of some finite set $\mu_1,\ldots\mu_k$, and replace the transmission of any $x\in\mathcal{X}$ by transmitting the index of the closest μ_i. The k-means objective can be viewed as a measure of the distortion created by such a transmission representation scheme.

The k-medoids objective function is similar to the k-means objective, except that it requires the cluster centroids to be members of the input set. The objective function is defined by

$$G_{\text{K-medoid}}((\mathcal{X},d),(C_1,\ldots,C_k)) = \min_{\mu_1,\ldots\mu_k\in\mathcal{X}} \sum_{i=1}^{k}\sum_{x\in C_i} d(x,\mu_i)^2.$$

The k-median objective function is quite similar to the k-medoids objective, except that the "distortion" between a data point and the centroid of its cluster is measured by distance, rather than by the square of the distance:

$$G_{\text{K-median}}((\mathcal{X},d),(C_1,\ldots,C_k)) = \min_{\mu_1,\ldots\mu_k\in\mathcal{X}} \sum_{i=1}^{k}\sum_{x\in C_i} d(x,\mu_i).$$

An example where such an objective makes sense is the *facility location* problem. Consider the task of locating k fire stations in a city. One can model houses as data points and aim to place the stations so as to minimize the average distance between a house and its closest fire station.

The previous examples can all be viewed as *center-based* objectives. The solution to such a clustering problem is determined by a set of cluster centers, and the clustering assigns each instance to the center closest to it. More generally, center-based objective is determined by choosing some monotonic function $f:\mathbb{R}_+\to\mathbb{R}_+$ and then defining

$$G_f((\mathcal{X},d),(C_1,\ldots C_k)) = \min_{\mu_1,\ldots\mu_k\in\mathcal{X}'} \sum_{i=1}^{k}\sum_{x\in C_i} f(d(x,\mu_i)),$$

where $\mathcal{X}'$ is either $\mathcal{X}$ or some superset of $\mathcal{X}$.

Some objective functions are not center based. For example, the *sum of in-cluster distances (SOD)*

$$G_{\text{SOD}}((\mathcal{X},d),(C_1,\ldots C_k)) = \sum_{i=1}^{k}\sum_{x,y\in C_i} d(x,y)$$

and the MinCut objective that we shall discuss in Section 22.3 are not center-based objectives.

22.2.1 The k-Means Algorithm

The k-means objective function is quite popular in practical applications of clustering. However, it turns out that finding the optimal k-means solution is often computationally infeasible (the problem is NP-hard, and even NP-hard to approximate to within some constant). As an alternative, the following simple iterative algorithm is often used, so often that, in many cases, the term k-means Clustering refers to the outcome of this algorithm rather than to the clustering that minimizes the k-means objective cost. We describe the algorithm with respect to the Euclidean distance function $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$.

k-Means

input: $\mathcal{X} \subset \mathbb{R}^n$; Number of clusters k
initialize: Randomly choose initial centroids $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$
repeat until convergence
  $\forall i \in [k]$ set $C_i = \{\mathbf{x} \in \mathcal{X} : i = \operatorname{argmin}_j \|\mathbf{x} - \boldsymbol{\mu}_j\|\}$
  (break ties in some arbitrary manner)
  $\forall i \in [k]$ update $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$

Lemma 22.1. *Each iteration of the k-means algorithm does not increase the k-means objective function (as given in Equation (22.1)).*

Proof. To simplify the notation, let us use the shorthand $G(C_1, \ldots, C_k)$ for the k-means objective, namely,

$$G(C_1, \ldots, C_k) = \min_{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \in \mathbb{R}^n} \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2. \tag{22.2}$$

It is convenient to define $\boldsymbol{\mu}(C_i) = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$ and note that $\boldsymbol{\mu}(C_i) = \operatorname{argmin}_{\boldsymbol{\mu} \in \mathbb{R}^n} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}\|^2$. Therefore, we can rewrite the k-means objective as

$$G(C_1, \ldots, C_k) = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}(C_i)\|^2. \tag{22.3}$$

Consider the update at iteration t of the k-means algorithm. Let $C_1^{(t-1)}, \ldots, C_k^{(t-1)}$ be the previous partition, let $\boldsymbol{\mu}_i^{(t-1)} = \boldsymbol{\mu}(C_i^{(t-1)})$, and let $C_1^{(t)}, \ldots, C_k^{(t)}$ be the new partition assigned at iteration t. Using the definition of the objective as given in Equation (22.2) we clearly have that

$$G(C_1^{(t)}, \ldots, C_k^{(t)}) \leq \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i^{(t)}} \|\mathbf{x} - \boldsymbol{\mu}_i^{(t-1)}\|^2. \tag{22.4}$$

In addition, the definition of the new partition $(C_1^{(t)}, \ldots, C_k^{(t)})$ implies that it minimizes the expression $\sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i^{(t-1)}\|^2$ over all possible partitions

$(C_1, \ldots, C_k)$. Hence,

$$\sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i^{(t)}} \|\mathbf{x} - \boldsymbol{\mu}_i^{(t-1)}\|^2 \leq \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i^{(t-1)}} \|\mathbf{x} - \boldsymbol{\mu}_i^{(t-1)}\|^2. \tag{22.5}$$

Using Equation (22.3) we have that the right-hand side of Equation (22.5) equals $G(C_1^{(t-1)}, \ldots, C_k^{(t-1)})$. Combining this with Equation (22.4) and Equation (22.5), we obtain that $G(C_1^{(t)}, \ldots, C_k^{(t)}) \leq G(C_1^{(t-1)}, \ldots, C_k^{(t-1)})$, which concludes our proof. □

While the preceding lemma tells us that the k-means objective is monotonically nonincreasing, there is no guarantee on the number of iterations the k-means algorithm needs in order to reach convergence. Furthermore, there is no nontrivial lower bound on the gap between the value of the k-means objective of the algorithm's output and the minimum possible value of that objective function. In fact, k-means might converge to a point which is not even a local minimum (see Exercise 22.2). To improve the results of k-means it is often recommended to repeat the procedure several times with different randomly chosen initial centroids (e.g., we can choose the initial centroids to be random points from the data).

22.3 SPECTRAL CLUSTERING

Often, a convenient way to represent the relationships between points in a data set $\mathcal{X} = \{x_1, \ldots, x_m\}$ is by a *similarity graph*; each vertex represents a data point x_i, and every two vertices are connected by an edge whose weight is their similarity, $W_{i,j} = s(x_i, x_j)$, where $W \in \mathbb{R}^{m,m}$. For example, we can set $W_{i,j} = \exp(-d(x_i, x_j)^2/\sigma^2)$, where $d(\cdot, \cdot)$ is a distance function and σ is a parameter. The clustering problem can now be formulated as follows: We want to find a partition of the graph such that the edges between different groups have low weights and the edges within a group have high weights.

In the clustering objectives described previously, the focus was on one side of our intuitive definition of clustering – making sure that points in the same cluster are similar. We now present objectives that focus on the other requirement – points separated into different clusters should be nonsimilar.

22.3.1 Graph Cut

Given a graph represented by a similarity matrix W, the simplest and most direct way to construct a partition of the graph is to solve the mincut problem, which chooses a partition $C_1, \ldots, C_k$ that minimizes the objective

$$\text{cut}(C_1, \ldots, C_k) = \sum_{i=1}^{k} \sum_{r \in C_i, s \notin C_i} W_{r,s}.$$

For $k = 2$, the mincut problem can be solved efficiently. However, in practice it often does not lead to satisfactory partitions. The problem is that in many cases, the solution of mincut simply separates one individual vertex from the rest of the graph.

Of course, this is not what we want to achieve in clustering, as clusters should be reasonably large groups of points.

Several solutions to this problem have been suggested. The simplest solution is to normalize the cut and define the normalized mincut objective as follows:

$$\text{RatioCut}(C_1, \ldots, C_k) = \sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{r \in C_i, s \notin C_i} W_{r,s}.$$

The preceding objective assumes smaller values if the clusters are not too small. Unfortunately, introducing this balancing makes the problem computationally hard to solve. Spectral clustering is a way to relax the problem of minimizing RatioCut.

22.3.2 Graph Laplacian and Relaxed Graph Cuts

The main mathematical object for spectral clustering is the graph Laplacian matrix. There are several different definitions of graph Laplacian in the literature, and in the following we describe one particular definition.

Definition 22.2 (Unnormalized Graph Laplacian). The *unnormalized graph Laplacian* is the $m \times m$ matrix $L = D - W$ where D is a diagonal matrix with $D_{i,i} = \sum_{j=1}^{m} W_{i,j}$. The matrix D is called the *degree matrix*.

The following lemma underscores the relation between RatioCut and the Laplacian matrix.

Lemma 22.3. *Let $C_1, \ldots, C_k$ be a clustering and let $H \in \mathbb{R}^{m,k}$ be the matrix such that*

$$H_{i,j} = \frac{1}{\sqrt{|C_j|}} \mathbb{1}_{[i \in C_j]}.$$

Then, the columns of H are orthonormal to each other and

$$\mathit{RatioCut}(C_1, \ldots, C_k) = \text{trace}(H^\top L H).$$

Proof. Let $\mathbf{h}_1, \ldots, \mathbf{h}_k$ be the columns of H. The fact that these vectors are orthonormal is immediate from the definition. Next, by standard algebraic manipulations, it can be shown that $\text{trace}(H^\top L H) = \sum_{i=1}^{k} \mathbf{h}_i^\top L \mathbf{h}_i$ and that for any vector $\mathbf{v}$ we have

$$\mathbf{v}^\top L \mathbf{v} = \frac{1}{2} \left(\sum_r D_{r,r} v_r^2 - 2 \sum_{r,s} v_r v_s W_{r,s} + \sum_s D_{s,s} v_s^2 \right) = \frac{1}{2} \sum_{r,s} W_{r,s} (v_r - v_s)^2.$$

Applying this with $\mathbf{v} = \mathbf{h}_i$ and noting that $(h_{i,r} - h_{i,s})^2$ is nonzero only if $r \in C_i, s \notin C_i$ or the other way around, we obtain that

$$\mathbf{h}_i^\top L \mathbf{h}_i = \frac{1}{|C_i|} \sum_{r \in C_i, s \notin C_i} W_{r,s}.$$

□

Therefore, to minimize RatioCut we can search for a matrix H whose columns are orthonormal and such that each $H_{i,j}$ is either 0 or $1/\sqrt{|C_j|}$. Unfortunately, this is an integer programming problem which we cannot solve efficiently. Instead, we

relax the latter requirement and simply search an orthonormal matrix $H \in \mathbb{R}^{m,k}$ that minimizes trace$(H^\top L H)$. As we will see in the next chapter about PCA (particularly, the proof of Theorem 23.2), the solution to this problem is to set U to be the matrix whose columns are the eigenvectors corresponding to the k minimal eigenvalues of L. The resulting algorithm is called Unnormalized Spectral Clustering.

22.3.3 Unnormalized Spectral Clustering

Unnormalized Spectral Clustering

Input: $W \in \mathbb{R}^{m,m}$; Number of clusters k
Initialize: Compute the unnormalized graph Laplacian L
Let $U \in \mathbb{R}^{m,k}$ be the matrix whose columns are the eigenvectors of L corresponding to the k smallest eigenvalues
Let $\mathbf{v}_1, \ldots, \mathbf{v}_m$ be the rows of U
Cluster the points $\mathbf{v}_1, \ldots, \mathbf{v}_m$ using k-means
Output: Clusters $C_1, \ldots, C_K$ of the k-means algorithm

The spectral clustering algorithm starts with finding the matrix H of the k eigenvectors corresponding to the smallest eigenvalues of the graph Laplacian matrix. It then represents points according to the rows of H. It is due to the properties of the graph Laplacians that this change of representation is useful. In many situations, this change of representation enables the simple k-means algorithm to detect the clusters seamlessly. Intuitively, if H is as defined in Lemma 22.3 then each point in the new representation is an indicator vector whose value is nonzero only on the element corresponding to the cluster it belongs to.

22.4 INFORMATION BOTTLENECK*

The information bottleneck method is a clustering technique introduced by Tishby, Pereira, and Bialek. It relies on notions from *information theory*. To illustrate the method, consider the problem of clustering text documents where each document is represented as a bag-of-words; namely, each document is a vector $\mathbf{x} = \{0,1\}^n$, where n is the size of the dictionary and $x_i = 1$ iff the word corresponding to index i appears in the document. Given a set of m documents, we can interpret the bag-of-words representation of the m documents as a joint probability over a random variable x, indicating the identity of a document (thus taking values in $[m]$), and a random variable y, indicating the identity of a word in the dictionary (thus taking values in $[n]$).

With this interpretation, the information bottleneck refers to the identity of a clustering as another random variable, denoted C, that takes values in $[k]$ (where k will be set by the method as well). Once we have formulated x, y, C as random variables, we can use tools from information theory to express a clustering objective. In particular, the information bottleneck objective is

$$\min_{p(C|x)} I(x;C) - \beta I(C;y) \,,$$

where $I(\cdot;\cdot)$ is the mutual information between two random variables,[1] β is a parameter, and the minimization is over all possible probabilistic assignments of points to clusters. Intuitively, we would like to achieve two contradictory goals. On one hand, we would like the mutual information between the identity of the document and the identity of the cluster to be as small as possible. This reflects the fact that we would like a strong compression of the original data. On the other hand, we would like high mutual information between the clustering variable and the identity of the words, which reflects the goal that the "relevant" information about the document (as reflected by the words that appear in the document) is retained. This generalizes the classical notion of minimal sufficient statistics[2] used in parametric statistics to arbitrary distributions.

Solving the optimization problem associated with the information bottleneck principle is hard in the general case. Some of the proposed methods are similar to the EM principle, which we will discuss in Chapter 24.

22.5 A HIGH LEVEL VIEW OF CLUSTERING

So far, we have mainly listed various useful clustering tools. However, some fundamental questions remain unaddressed. First and foremost, what is clustering? What is it that distinguishes a *clustering* algorithm from any arbitrary function that takes an input space and outputs a partition of that space? Are there any basic properties of clustering that are independent of any specific algorithm or task?

One method for addressing such questions is via an axiomatic approach. There have been several attempts to provide an axiomatic definition of clustering. Let us demonstrate this approach by presenting the attempt made by Kleinberg (2003).

Consider a clustering function, F, that takes as input any finite domain $\mathcal{X}$ with a dissimilarity function d over its pairs and returns a partition of $\mathcal{X}$.

Consider the following three properties of such a function:

Scale Invariance (SI) For any domain set $\mathcal{X}$, dissimilarity function d, and any $\alpha > 0$, the following should hold: $F(\mathcal{X}, d) = F(\mathcal{X}, \alpha d)$ (where $(\alpha d)(x, y) \stackrel{\text{def}}{=} \alpha\, d(x, y)$).

Richness (Ri) For any finite $\mathcal{X}$ and every partition $C = (C_1, \ldots C_k)$ of X (into nonempty subsets) there exists some dissimilarity function d over $\mathcal{X}$ such that $F(\mathcal{X}, d) = C$.

Consistency (Co) If d and d' are dissimilarity functions over $\mathcal{X}$, such that for every $x, y \in \mathcal{X}$, if x, y belong to the same cluster in $F(\mathcal{X}, d)$ then $d'(x, y) \leq d(x, y)$ and if x, y belong to different clusters in $F(\mathcal{X}, d)$ then $d'(x, y) \geq d(x, y)$, then $F(\mathcal{X}, d) = F(\mathcal{X}, d')$.

[1] That is, given a probability function, p over the pairs (x, C), $I(x;C) = \sum_a \sum_b p(a, b) \log\left(\frac{p(a,b)}{p(a)p(b)}\right)$, where the sum is over all values x can take and all values C can take.

[2] A sufficient statistic is a function of the data which has the property of sufficiency with respect to a statistical model and its associated unknown parameter, meaning that "no other statistic which can be calculated from the same sample provides any additional information as to the value of the parameter." For example, if we assume that a variable is distributed normally with a unit variance and an unknown expectation, then the average function is a sufficient statistic.

A moment of reflection reveals that the Scale Invariance is a very natural requirement – it would be odd to have the result of a clustering function depend on the units used to measure between-point distances. The Richness requirement basically states that the outcome of the clustering function is fully controlled by the function d, which is also a very intuitive feature. The third requirement, Consistency, is the only requirement that refers to the basic (informal) definition of clustering – we wish that similar points will be clustered together and that dissimilar points will be separated to different clusters, and therefore, if points that already share a cluster become more similar, and points that are already separated become even less similar to each other, the clustering function should have even stronger "support" of its previous clustering decisions.

However, Kleinberg (2003) has shown the following "impossibility" result:

Theorem 22.4. *There exists no function, F, that satisfies all the three properties: Scale Invariance, Richness, and Consistency.*

Proof. Assume, by way of contradiction, that some F does satisfy all three properties. Pick some domain set $\mathcal{X}$ with at least three points. By Richness, there must be some d_1 such that $F(\mathcal{X}, d_1) = \{\{x\} : x \in \mathcal{X}\}$ and there also exists some d_2 such that $F(\mathcal{X}, d_2) \neq F(\mathcal{X}, d_1)$.

Let $\alpha \in \mathbb{R}_+$ be such that for every $x, y \in \mathcal{X}$, $\alpha d_2(x, y) \geq d_1(x, y)$. Let $d_3 = \alpha d_2$. Consider $F(\mathcal{X}, d_3)$. By the Scale Invariance property of F, we should have $F(\mathcal{X}, d_3) = F(\mathcal{X}, d_2)$. On the other hand, since all distinct $x, y \in \mathcal{X}$ reside in different clusters w.r.t. $F(\mathcal{X}, d_1)$, and $d_3(x, y) \geq d_1(x, y)$, the Consistency of F implies that $F(\mathcal{X}, d_3) = F(\mathcal{X}, d_1)$. This is a contradiction, since we chose d_1, d_2 so that $F(\mathcal{X}, d_2) \neq F(\mathcal{X}, d_1)$. □

It is important to note that there is no single "bad axiom" there is no single "bad property" among the three properties. For every pair of the three axioms, there exist natural clustering functions that satisfy the two properties in that pair (one can even construct such examples just by varying the stopping criteria for the Single Linkage clustering function). On the other hand, Kleinberg shows that any clustering algorithm that minimizes any center-based objective function inevitably fails the consistency property (yet, the k-sum-of-in-cluster-distances minimization clustering does satisfy Consistency).

The Kleinberg impossibility result can be easily circumvented by varying the properties. For example, if one wishes to discuss clustering functions that have a fixed number-of-clusters parameter, then it is natural to replace Richness by k-Richness (namely, the requirement that every partition of the domain into k subsets is attainable by the clustering function). k-Richness, Scale Invariance and Consistency all hold for the k-means clustering and are therefore consistent. Alternatively, one can relax the Consistency property. For example, say that two clusterings $C = (C_1, \ldots C_k)$ and $C' = (C'_1, \ldots C'_l)$ are *compatible* if for every clusters $C_i \in C$ and $C'_j \in C'$, either $C_i \subseteq C'_j$ or $C'_j \subseteq C_i$ or $C_i \cap C'_j = \emptyset$ (it is worthwhile noting that for every dendrogram, every two clusterings that are obtained by trimming that dendrogram are compatible). "Refinement Consistency" is the requirement that, under the assumptions of the Consistency property, the new clustering $F(\mathcal{X}, d')$ is compatible with the old clustering $F(\mathcal{X}, d)$. Many common clustering functions satisfy this

requirement as well as Scale Invariance and Richness. Furthermore, one can come up with many other, different, properties of clustering functions that sound intuitive and desirable and are satisfied by some common clustering functions.

Furthermore, one can come up with many other, different, properties of clustering functions that sound intuitive and desirable and are satisfied by some common clustering functions.

There are many ways to interpret these results. We suggest to view it as indicating that there is no "ideal" clustering function. Every clustering function will inevitably have some "undesirable" properties. The choice of a clustering function for any given task must therefore take into account the specific properties of that task. There is no generic clustering solution, just as there is no classification algorithm that will learn every learnable task (as the No-Free-Lunch theorem shows). Clustering, just like classification prediction, must take into account some prior knowledge about the specific task at hand.

22.6 SUMMARY

Clustering is an unsupervised learning problem, in which we wish to partition a set of points into "meaningful" subsets. We presented several clustering approaches including linkage-based algorithms, the k-means family, spectral clustering, and the information bottleneck. We discussed the difficulty of formalizing the intuitive meaning of clustering.

22.7 BIBLIOGRAPHIC REMARKS

The k-means algorithm is sometimes named Lloyd's algorithm, after Stuart Lloyd, who proposed the method in 1957. For a more complete overview of spectral clustering we refer the reader to the excellent tutorial by Von Luxburg (2007). The information bottleneck method was introduced by Tishby, Pereira, and Bialek (1999). For an additional discussion on the axiomatic approach see Ackerman and Ben-David (2008).

22.8 EXERCISES

22.1 **Suboptimality of k-Means:** For every parameter $t > 1$, show that there exists an instance of the k-means problem for which the k-means algorithm (might) find a solution whose k-means objective is at least $t \cdot \text{OPT}$, where OPT is the minimum k-means objective.

22.2 **k-Means Might Not Necessarily Converge to a Local Minimum:** Show that the k-means algorithm might converge to a point which is not a local minimum. *Hint*: Suppose that $k = 2$ and the sample points are $\{1, 2, 3, 4\} \subset \mathbb{R}$ suppose we initialize the k-means with the centers $\{2, 4\}$; and suppose we break ties in the definition of C_i by assigning i to be the smallest value in $\text{argmin}_j \|\mathbf{x} - \boldsymbol{\mu}_j\|$.

22.3 Given a metric space $(\mathcal{X}, d)$, where $|\mathcal{X}| < \infty$, and $k \in \mathbb{N}$, we would like to find a partition of $\mathcal{X}$ into $C_1, \ldots, C_k$ which minimizes the expression

$$G_{\text{k-diam}}((\mathcal{X}, d), (C_1, \ldots, C_k)) = \max_{j \in [d]} \text{diam}(C_j),$$

where $\text{diam}(C_j) = \max_{x,x' \in C_j} d(x,x')$ (we use the convention $\text{diam}(C_j) = 0$ if $|C_j| < 2$).

Similarly to the k-means objective, it is NP-hard to minimize the *k-diam* objective. Fortunately, we have a very simple approximation algorithm: Initially, we pick some $x \in \mathcal{X}$ and set $\mu_1 = x$. Then, the algorithm iteratively sets

$$\forall j \in \{2,\ldots,k\},\ \mu_j = \operatorname*{argmax}_{x \in X} \min_{i \in [j-1]} d(x,\mu_i).$$

Finally, we set

$$\forall i \in [k],\ C_i = \{x \in X : i = \operatorname*{argmin}_{j \in [k]} d(x,\mu_j)\}.$$

Prove that the algorithm described is a 2-approximation algorithm. That is, if we denote its output by $\hat{C}_1,\ldots,\hat{C}_k$, and denote the optimal solution by $C_1^*,\ldots,C_k^*$, then,

$$G_{\text{k-diam}}((\mathcal{X},d),(\hat{C}_1,\ldots,\hat{C}_k)) \leq 2 \cdot G_{\text{k-diam}}((\mathcal{X},d),(C_1^*,\ldots,C_k^*)).$$

Hint: Consider the point μ_{k+1} (in other words, the next center we would have chosen, if we wanted $k+1$ clusters). Let $r = \min_{j \in [k]} d(\mu_j,\mu_{k+1})$. Prove the following inequalities

$$G_{\text{k-diam}}((\mathcal{X},d),(\hat{C}_1,\ldots,\hat{C}_k)) \leq 2r$$
$$G_{\text{k-diam}}((X,d),(C_1^*,\ldots,C_k^*)) \geq r.$$

22.4 Recall that a clustering function, F, is called Center-Based Clustering if, for some monotonic function $f : \mathbb{R}_+ \to \mathbb{R}_+$, on every given input $(\mathcal{X},d)$, $F(\mathcal{X},d)$ is a clustering that minimizes the objective

$$G_f((\mathcal{X},d),(C_1,\ldots C_k)) = \min_{\mu_1,\ldots\mu_k \in \mathcal{X}'} \sum_{i=1}^{k} \sum_{x \in C_i} f(d(x,\mu_i)),$$

where $\mathcal{X}'$ is either $\mathcal{X}$ or some superset of $\mathcal{X}$.

Prove that for every $k > 1$ the k-diam clustering function defined in the previous exercise is not a center-based clustering function.

Hint: Given a clustering input $(\mathcal{X},d)$, with $|\mathcal{X}| > 2$, consider the effect of adding many close-by points to some (but not all) of the members of $\mathcal{X}$, on either the k-diam clustering or any given center-based clustering.

22.5 Recall that we discussed three clustering "properties": Scale Invariance, Richness, and Consistency. Consider the Single Linkage clustering algorithm.

1. Find which of the three properties is satisfied by Single Linkage with the Fixed Number of Clusters (any fixed nonzero number) stopping rule.
2. Find which of the three properties is satisfied by Single Linkage with the Distance Upper Bound (any fixed nonzero upper bound) stopping rule.
3. Show that for any pair of these properties there exists a stopping criterion for Single Linkage clustering, under which these two axioms are satisfied.

22.6 Given some number k, let k-Richness be the following requirement:
For any finite $\mathcal{X}$ and every partition $C = (C_1,\ldots C_k)$ of $\mathcal{X}$ (into nonempty subsets) there exists some dissimilarity function d over $\mathcal{X}$ such that $F(\mathcal{X},d) = C$.

Prove that, for every number k, there exists a clustering function that satisfies the three properties: Scale Invariance, k-Richness, and Consistency.