Partition-Based Clustering with k-Means

Chapter · February 2016

DOI: 10.1007/978-3-319-21903-5_7

1 author:

Frank Nielsen
Sony Computer Science Laboratories, Inc.
512 PUBLICATIONS **5,958** CITATIONS

Undergraduate Topics in Computer Science

Frank Nielsen

Introduction to HPC with MPI for Data Science

UTiCS

Springer

7 Partition-based clustering with k-means

A concise summary is provided at the end of this chapter, in §7.11.

7.1 Exploratory data analysis and clustering

Nowadays, huge size data-sets are commonly publicly available, and it becomes increasingly important to efficiently process them to discover worthwhile structures (or "patterns") in those seas of data. *Exploratory* data analysis is concerned with this challenge of finding such structural information without any prior knowledge: In this case, those techniques that consist in learning from data without prior knowledge are called generically *unsupervised machine learning*.

Let $X = \{x_1, ..., x_n\}$ denote a data-set like a collection of images (often static, that is given once for all when we begin with the analysis). We seek for compact subsets of data, called clusters, that represent *categories* of data (say, the *cluster* of the car images or the cluster of the cat images, etc.). Each datum $x_i \in \mathbb{X}$ (with $\mathbb{X}$ denoting the space of data, usually $\mathbb{X} \subset \mathbb{R}^d$) is described as an *attribute vector* $x_i = (x_i^1, ..., x_i^d)$, called a *feature vector*. We adopt the following notation x_i^j (meaning $x_i^{(j)}$) to describe the j-th coordinate of vector x_i. Vector attributes can either be *numerical quantities* or *categorical values* (that is qualitative attributes) like the fixed set of words of a dictionary, or *ordered categorical data* (like the ordered ranking A < B < C < D < E), or a mix of those different data types.

Exploratory analysis differs from *supervised classification* that consists in a first stage to learn a *classifier function* $C(\cdot)$ from labeled data of a *training data set* $Z = \{(x_1, y_1), ..., (x_n, y_n)\}$, with the x_i's the attributes and the y_i's the class labels, in order in a second stage to classify new unlabeled observations x_j belonging to a *testing set*: $\hat{y}_j = C(x_j)$. The hat notation in $\hat{y}_j$ indicates that one has *estimated* the class, a so-called *inference task* performed from a training data set.

Clustering is a set of techniques that consists in detecting subsets of data that define groups or clusters. Those groups should ideally represent *semantic categories* of data: For example, the flower groups organiwed by species from a database of flower images. One such famous public data-set is available from the *UCI repository*[1] as filename `Iris`[2]: It contains $n = 150$ numerical data in dimension 4 (with attributes describing the length and the width of both the sepals and petals, in centimeters), classified in $k = 3$ botanical groups: Setosa iris, Virginica iris, and Versicolor iris.

To summarize, classification enables to label new observations while clustering allows one to discover those classes as clusters.

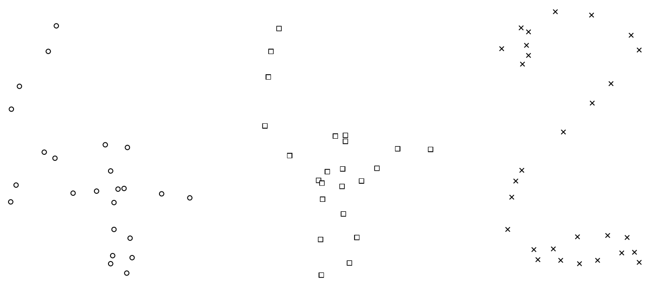


Figure 7.1 Exploratory analysis consists in finding intrinsic structures in data sets like groups of data called cluster. Clustering is a set of techniques that seek to find homogeneous clusters in data-sets. In this 2D toy example, the Human eye perceives three well-formed clusters for the digits: '4', '4', '2'. In practice, data-sets are living in high dimensions, and thus cannot be visually inspected: Therefore we require clustering algorithms to automatically find those groups.

[1] Available online at `https://archive.ics.uci.edu/ml/datasets.html`
[2] `http://en.wikipedia.org/wiki/Iris_flower_data_set`

7.1.1 Hard clustering: Partitioning data sets

Partition-based clustering consists in dividing a data set $X = \{x_1, ..., x_n\}$ into k *homogeneous groups* $G_1 \subset X, ..., G_k \subset X$ (the non-overlapping clusters G_i) such that we have:

$$\begin{aligned} X &= \cup_{i=1}^k G_i, \quad \forall\ i \neq j,\ G_i \cap G_j = \emptyset, \\ X &:= \uplus_{i=1}^k G_i \end{aligned}$$

Notation $a := b$ indicates that the equality sign should be understood by definition (that is, it is not an equality resulting from some mathematical calculation). Thus a data element (datum) is allocated to a unique group $G_{l(x_i)}$: Partition-based clustering is a *hard clustering technique*, and differentiates itself from other *soft clustering techniques* that gives a positive membership weight $l_{i,j} > 0$ for all the x_i's and the groups $G_{l(x_i)}$'s with $\sum_{j=1}^k l_{i,j} = 1$ (normalization constraint): $l_{i,j} = 1$ if and only if (iff.) $j = l(x_i)$. We denote by $L = [l_{i,j}]$ the *membership matrix* of size $n \times k$.

7.1.2 Cost functions and model-based clustering

Finding a good partitioning of the data $X = \uplus_{i=1}^k G_i$ requires to be able to evaluate the clustering *fitness* of partitions. However, we often proceed the other way around! From a given *cost function*, we seek an efficient algorithm that partitions X by minimizing this prescribed cost function. A generic cost function $e_k(\cdot;\cdot)$ (also synonymously called *energy function*, *loss function* or *objective function*) is written as the sum of the costs of each group as follows:

$$e_k(X; G_1, ..., G_k) = \sum_{i=1}^k e_1(G_i),$$

with $e_1(G)$ the cost function for a single group.

We can also associate for each group G_i a *model* c_i that defines the "center" of that cluster. The collection of centers, the c_i's, are called the *prototypes*, and those prototypes allow one to define a distance between any data $x \in X$ and any cluster G (with corresponding prototype c) as follows:

$$D_M(x, G) = D(x, c).$$

Function $D_M(x, G)$ denotes the distance between an element x and a cluster using the prototype of that cluster. Function $D(p, q)$ is a *base distance* to properly define according to nature of the data set. That is, we have $D_M(x, G) = D(x, c)$ where c is the prototype of G.

Given the set of k prototypes $C = \{c_1, ..., c_k\}$, one can define the overall cost of a partition-based clustering by:

$$e_k(X; C) = \sum_{i=1}^{n} \min_{j \in \{1,\dots,k\}} D(x_i, c_j),$$

and the cost of a single cluster is defined by $e_1(G, c) = \sum_{x \in G} D(x, c)$. Model-based clustering with a single center associated to each cluster induces a partition of the data set X: $G(C) = \uplus_{j=1}^{k} G_j$, with $G_j = \{x_i \in X \ : \ D(x_i, c_j) \leq D(x_i, c_l), \ \forall \ l \in \{1, ..., k\}\}$.

There exists many clustering cost/loss functions that gives rise to many different kinds of partitions. Next, we shall introduce the most celebrated such a function called *k-means*, and explain why the minimization of this loss function provides good clustering partitions in practice.

7.2 The *k*-means objective function

The *k-means cost function* asks to minimize the sum of squared Euclidean distances of data points to their closest prototype centers:

$$e_k(X; C) = \sum_{i=1}^{n} \min_{j \in \{1,\dots,k\}} \|x_i - c_j\|^2.$$

Although that the squared Euclidean distance $D(x, c) = \|x - c\|^2$ is a symmetric dissimilarity measure equals to zero if and only if $x = c$, it is *not* a metric because it fails to satisfy the triangular inequalities of the ordinary Euclidean distance: $\|x - c\|_2 = \sqrt{\sum_{j=1}^{d} (x^i - c^j)^2}$. In fact, there is a good reason to choose the squared Euclidean distance instead of the Euclidean distance: Indeed, the cost of a single cluster $e_1(G) = e_1(G, c)$ is minimized when we choose for the cluster prototype its center of mass c, called the *centroid*:

$$c(G) := \mathrm{argmin}_c \sum_{x \in G} \|x - c\|^2 = \frac{1}{|G|} \sum_{x \in G} x,$$

where $|G|$ denotes the cardinality of G, that is the number of elements contained in group G. We use the following notation $\mathrm{argmin}_x f(x)$ to denote the argument that yields the minimum in case this minimum value is unique.[3]

[3] Otherwise, we can choose the “smallest” x that yields the minimum value according to some lexicographic order on $\mathbb{X}$.

Thus the minimal cost is $e_1(G,c) = \sum_{x\in G} \|x - c(G)\|^2 := v(G)$, the normalized variance of cluster G. Indeed, the normalized variance of X is defined in statistics as:

$$v(X) = \frac{1}{n}\sum_{i=1}^{n} \|x_i - \bar{x}\|^2,$$

with $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ the center of mass. Technically speaking, we often meet in the literature the unbiased variance formula $v_{\text{unbiased}}(X) = \frac{1}{n-1}\sum_{i=1}^{n}\|x_i - \bar{x}\|^2$, but for fixed n the minimizer of the biased/unbiased variance does not change.

We can rewrite the variance of a d-dimensional point cloud X as:

$$\boxed{v(X) = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^2\right) - \bar{x}^\top \bar{x}}$$

This formula is mathematically the same as the variance for a random variable X:

$$\mathbb{V}[X] = \mathbb{E}[(X - \mu(X))^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$$

where $\mu(X) = \mathbb{E}[X]$ denotes the expectation of the random variable X.

We can define a positive weight attribute $w_i = w(x_i) > 0$ for each element x_i of X such that $\sum_{i=1}^{n} w_i = 1$ (data weights normalized).

The following theorem characterizes the prototype c_1 as the center for a single cluster (case $k = 1$ with $X = G_1$):

Theorem 3

Let $X = \{(w_1, x_1), ..., (w_n, x_n)\} \subset \mathbb{R}^d$ be a weighted data-set with $w_i > 0$ and $\sum_{i=1}^{n} w_i = 1$. The center c that minimizes the weighted variance $v(X) = \sum_{i=1}^{n} w_i\|x_i - c\|^2$ is the unique barycenter: $c = \bar{x} = \sum_{i=1}^{n} w_i x_i$.

Proof

Let $\langle x, y\rangle$ denote the *scalar product*: $\langle x, y\rangle = x^\top y = \sum_{j=1}^{d} x^j y^j = \langle y, x\rangle$. The scalar product is a symmetric bi-linear form: $\langle \lambda x + b, y\rangle = \lambda\langle x, y\rangle + \langle b, y\rangle$ for $\lambda \in \mathbb{R}$. Now, the squared Euclidean distance $D(x,y) = \|x-y\|^2$ can be rewritten using scalar producs as $D(x,y) = \langle x - y, x - y\rangle = \langle x, x\rangle - 2\langle x, y\rangle + \langle y, y\rangle$.

We seek to minimize $\min_{c\in\mathbb{R}^d}\sum_{i=1}^n w_i\langle x_i - c, x_i - c\rangle$. We can mathematically rewrite this optimization problem as:

$$\min_{c\in\mathbb{R}^d} \quad \sum_{i=1}^n w_i\langle x_i - c, x_i - c\rangle$$

$$\min_{c\in\mathbb{R}^d} \quad \sum_{i=1}^n w_i(\langle x_i, x_i\rangle - 2\langle x_i, c\rangle + \langle c, c\rangle)$$

$$\min_{c\in\mathbb{R}^d} \quad \left(\sum_{i=1}^n w_i\langle x_i, x_i\rangle\right) - 2\left\langle \sum_{i=1}^n w_i x_i, c\right\rangle + \langle c, c\rangle$$

We can remove the term $\sum_{i=1}^n w_i\langle x_i, x_i\rangle$ from the minimization since it is independent of c. Thus we seek to minimize equivalently:

$$\min_{c\in\mathbb{R}^d} E(c) := -2\left\langle \sum_{i=1}^n w_i x_i, c\right\rangle + \langle c, c\rangle.$$

A *convex function* $f(x)$ satisfies $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$ for any $\alpha \in [0,1]$. It is a *strictly convex function* iff. $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y)$ for any $\alpha \in (0,1)$. Figure 7.2 plots the graph of a strictly convex function. Note that the *epigraph* defined as the geometric object $\mathcal{F} = \{(x, f(x)) \ : \ x \in \mathbb{R}\}$ is a geometric convex object. A *geometric convex object* satisfies the property that any line segment joining two points of the object shall fully lie inside the object.

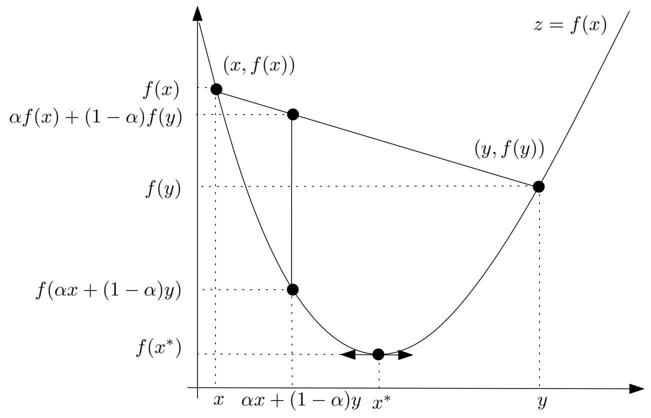


Figure 7.2 Plot of a strictly convex function satisfying $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y)$ for any $\alpha \in (0,1)$.

For univariate convex functions, there exists at most one global minimum x^* (for example, $\exp(-x)$ is strictly convex without a minimum), and it can be found by setting the derivative to zero: $f'(x^*) = 0$. For a multi-variate real-valued function, we denote by $\nabla_x F(x)$ its *gradient* (the vector of partial derivatives), and by $\nabla_x^2 F(x)$ the *Hessian matrix* (of second-order derivatives). A smooth function F is strictly convex if and only if $\nabla^2 F \succ 0$ where $M \succ 0$ denotes that the matrix M is positive-definite: $\forall x \neq 0,\ x^\top M x > 0$. A strictly convex function admits at most a global unique minimum x^* such that $\nabla F(x^*) = 0$.

We get the following d partial derivatives to set to zero:

$$\frac{\mathrm{d}}{\mathrm{d}c^j} E(c) = -2\sum_{i=1}^n w_i x_i^j + 2c^j, \quad \forall j \in \{1, ..., d\},$$

Consider the d^2 second derivatives (for proving the convexity of the objective function) as:

$$\frac{\mathrm{d}^2}{\mathrm{d}c^j c^l} E(c) = 2, \quad \text{for } l = j, \forall j \in \{1, ..., d\}.$$

The cost function $E(c)$ is strictly convex and admits a unique minimum. This minimum is obtained by zeroing all partial derivatives:

$$\frac{\mathrm{d}}{\mathrm{d}c^j} E(c) = 0 \Leftrightarrow c^j = \sum_{i=1}^n w_i x_i^j.$$

Thus we have shown that the minimizers of the weighted sum of squared Euclidean distance of the center to the points is the unique barycenter:

$$c = \bar{x} = \sum_{i=1}^n w_i x_i.$$

The centroid is also called *isobarycenter* when $w_i = \frac{1}{n}$. □

If instead of choosing the squared Euclidean distance, we had chosen the ordinary Euclidean distance, one obtains the so-called *Fermat-Weber point* that generalizes the notion of median. It is thus also called the *geometric median*.[4] Although the Fermat-Weber point is unique and often used in operations research for facility location problems, it does not admit a closed-form solution, but can be arbitrarily finely approximated. The *k-median clustering* is the clustering obtained by minimizing the cost function $\min_C \sum_{i=1}^n \min_{j\in\{1,\ldots,k\}} \|x_i - c_j\|$ (observe that the squared Euclidean distance of k-means has been replaced by the regular Euclidean distance). Note that the obtained partitions from k-means and k-medians can be very different from each other. Indeed, the centroid location can be different to the median for a single

[4] `http://en.wikipedia.org/wiki/Geometric_median`

cluster. Moreover, centroids can be easily corrupted by adding a single outlier point. We say that the *breakdown point* of the centroid is 0: A single outlier p_0 diverging to infinity will impact the centroid to be diverging to infinity too. But the median is provably more robust since it requires $\lfloor \frac{n}{2} \rfloor$ outliers (that is, about 50% of outliers) to steer the median point to ∞. Therefore k-median clustering is often preferred when there are many outliers in data-sets.

Let us remark that finding the center of a single cluster is a particular case of clustering with $k = 1$ cluster. With the squared Euclidean distance cost, we find that the center is the mean of the attributes, hence its naming: k-means. Figure 7.3 displays the clustering result on a given data-set. This figure has been produced using the following code in the *R language*[5]:

WWW source code: `Example-kMeans.R`

```
# filename: Example-kMeans.R
# k-means clustering using the R language
N <- 100000
x <- matrix(0, N, 2)
x[seq(1,N,by=4),]  <- rnorm(N/2)
x[seq(2,N,by=4),]  <- rnorm(N/2, 3, 1)
x[seq(3,N,by=4),]  <- rnorm(N/2, -3, 1)
x[seq(4,N,by=4),1] <- rnorm(N/4, 2, 1)
x[seq(4,N,by=4),2] <- rnorm(N/4, -2.5, 1)
start.kmeans <- proc.time()[3]
ans.kmeans <- kmeans(x, 4, nstart=3, iter.max=10,
    algorithm="Lloyd")
ans.kmeans$centers
end.kmeans <- proc.time()[3]
end.kmeans - start.kmeans
these <- sample(1:nrow(x), 1000)
plot(x[these,1], x[these,2], pch="+",xlab="x", ylab="y")
title(main="Clustering", sub="(globular shapes of
    clusters)", xlab="x", ylab="y")
points(ans.kmeans$centers, pch=19, cex=2, col=1:4)
```

[5] Freely available online at `https://www.r-project.org/`

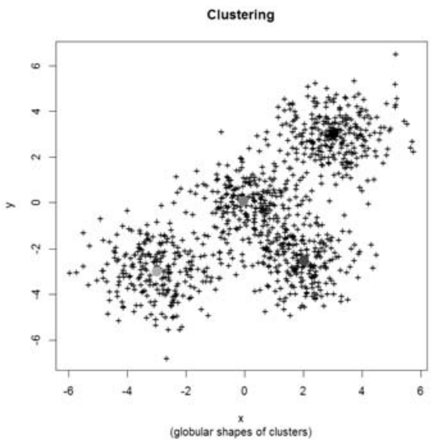


Figure 7.3 The k-means cost function tend to find globular-shaped clusters that minimize the weighted sum of the cluster variances. k-Means clustering is a model-based clustering where each cluster is associated to a prototype: its center of mass, or centroid. Here, we have choosen $k = 4$ groups for the k-means: Cluster prototypes, centroids, are illustrated with large disks.

7.2.1 Rewriting the *k*-means cost function for a dual interpretation of clustering: group intra-cluster or separate inter-cluster data

The k-means cost function seeks compact globular clusters of small variances. Indeed, the cost function can be reinterpreted as the minimization of the weighted sum of cluster variances as follows:

$$\min_{C=\{c_1,\ldots,c_k\}} \sum_{i=1}^{n} \min_{j\in\{1,\ldots,k\}} w_i \|x_i - c_j\|^2,$$

$$\min_{C=\{c_1,\ldots,c_k\}} \sum_{j=1}^{k} \sum_{x\in G_j} w(x)\|x - c_j\|^2$$

$$\min_{C=\{c_1,\ldots,c_k\}} \sum_{j=1}^{k} W_j v(G_j),$$

where $W_j := \sum_{x\in G_j} w(x)$ denotes the cumulative weight of the elements in cluster G_j (see exercise 7.12).

We can also show that clustering data into homogeneous groups correspond equivalently to separate data of X into groups: Indeed, let $A :=$

$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\|x_i - x_j\|^2$ denote the *constant* that is the sum of the inter-point squared Euclidean distances (fixed for a given data-set, and independent of k). For a given partition, we can decompose A into two terms: the sum of the intra-distances inside a same cluster, and the sum of the inter-distance among two distinct clusters:

$$A = \sum_{i=1}^{l}\left(\sum_{x_i, x_j \in G_l}\|x_i - x_j\|^2 + \sum_{x_i \in G_l, x_j \notin G_l}\|x_i - x_j\|^2\right).$$

Thus to minimize the sum of the intra-cluster squared Euclidean distances $\sum_{i=1}^{l}\sum_{x_i,x_j\in G_l}\|x_i - x_j\|^2$ is equivalent to maximize the sum of the inter-cluster squared Euclidean distances since A is a constant (for a given X):

$$\begin{aligned}
&\min_C && \sum_{i=1}^{l}\sum_{x_i,x_j\in G_l}\|x_i - x_j\|^2 \\
=&\min_C && A - \sum_{i=1}^{l}\sum_{x_i\in G_l, x_j\notin G_l}\|x_i - x_j\|^2 \\
\equiv&\max_C && \sum_{i=1}^{l}\sum_{x_i\in G_l, x_j\notin G_l}\|x_i - x_j\|^2
\end{aligned}$$

Therefore, we have a dual description to define a good clustering:

– *cluster* data into *homogeneous groups* in order to minimize the weighted sum of cluster variances, or

– *separate* data in order to maximize the inter-cluster squared Euclidean distances.

7.2.2 Complexity and tractability of the *k*-means optimization problem

Finding the minimum of a k-means cost function is a *NP-hard problem* as soon as the dimension $d > 1$ and the number of clusters $k > 1$. When $k = 1$, we have shown that we can compute the optimal solution (the centroid) in linear time (computing the mean of the group). When $d = 1$, we can compute an optimal k-means solution using *dynamic programming*: Using $O(nk)$ memory, we can solve the k-means for n scalar values in time $O(n^2k)$ (see the exercises at the end of this chapter for further details).

Theorem 4 (k-means complexity)

Finding a partition that minimizes the k-means cost function is NP-hard when $k > 1$ and $d > 1$. When $d = 1$, we can solve for the exact k-means using dynamic programming in $O(n^2 k)$ time using $O(nk)$ memory.

We quickly recall that P is the class of decision problems (that is, answering yes/no questions) that can be solved in polynomial time, and NP is the class of problems for which one can verify the solution in *polynomial time* (like for example, 3-SAT[6]). The NP-complete class is that class of problems that can be solved one from another by using a polynomial-time reduction scheme: $X \propto_{\text{polynomial}} Y, \forall Y \in \text{NP}$. The NP-hard class is the class of problems, not necessarily in NP, such that $\exists Y \in \text{NP} - \text{Complete} \propto_{\text{polynomial}} X$.

Since the k-means problem is theoretically NP-hard, we seek efficient heuristics to approximate the cost function. We distinguish two classes of such heuristics:

1. *global heuristics* that do not depend on initialization, and
2. *local heuristics* that iteratively starts from a solution (a partition) and iteratively improves this partition using "pivot rules."

Of course, one need to initialize local heuristics with a global heuristic. This yields many strategies for obtaining in practice a good k-means clustering! Finding novel k-means heuristics is still an active research topic 50 years after its inception!

7.3 Lloyd's batched k-means local heuristic

We now present the celebrated Lloyd's heuristic (1957) that consists from a given initialization to iteratively repeat until convergence the following two steps:

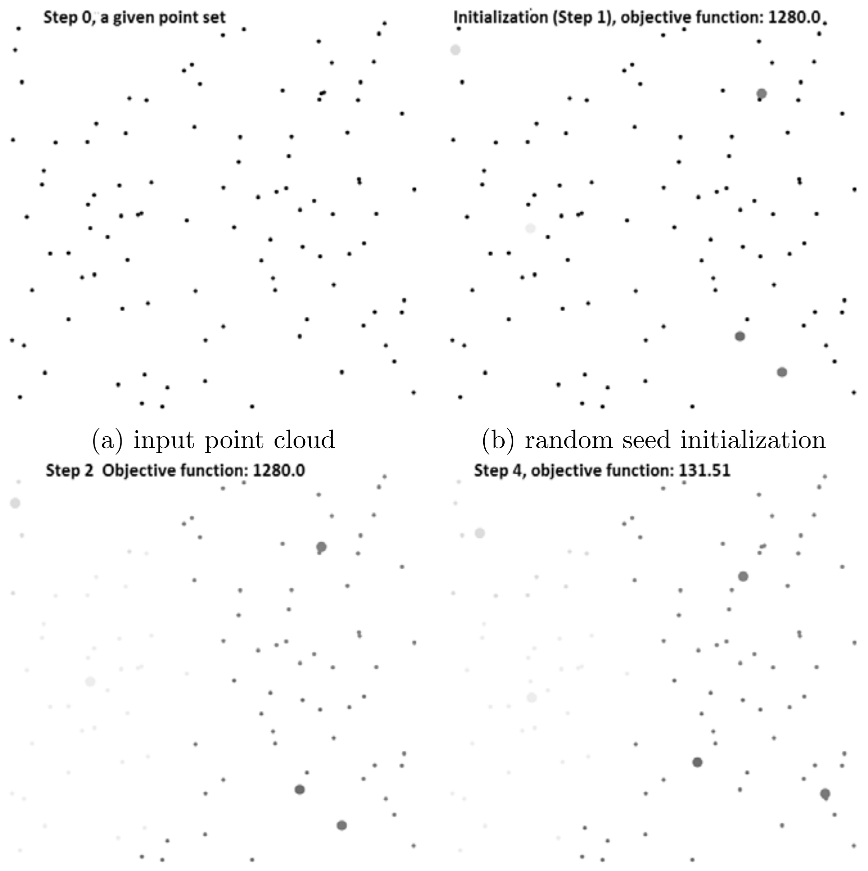
Assign points to clusters. For all $x_i \in X$, let $l_i = \text{argmin}_l \|x_i - c_l\|^2$, and define the k cluster groups as $G_j = \{x_i : l_i = j\}$ with $n_j = |G_j|$, the number of elements of X falling into the j-th cluster.

Update centers. For all $j \in \{1, ..., k\}$, update the centers to their cluster centroids : $c_j = \frac{1}{n_j} \sum_{x \in G_j} x$ (or the barycenters $c_j = \frac{1}{\sum_{x \in G_j} w(x)} \sum_{x \in G_j} w(x) x$

[6] The 3-*SAT problem* consists in answering whether a boolean formula with n clauses of 3 literals can be satisfiable or not. 3-SAT is a famouns NP-complete problem (Cook's theorem, 1971), a corner stone of theoretical computer science.

for weighted data-sets).

Figure 7.4 illustrates a few iterations of Lloyd's algorithm.



(a) input point cloud (b) random seed initialization

(c) assign points to clusters (d) new centers = centroids

Figure 7.4 Illustration of Lloyd's k-means algorithm: (a) input data set, (b) random initialization of cluster centers, (c) assigning points to clusters, (d) center relocations, etc. until the algorithm convergences into a local minimum of the cost function.

Theorem 5

Lloyd's k-means heuristics converge monotonically into a local minimum after a finite number of iterations, upper bounded by $\binom{n}{k}$.

Proof

Let $G_1^{(t)}, ..., G_k^{(t)}$ denote the partition of X at time t, with cost $e_k(X, C_t)$. Let $G(C_t) = \uplus_{j=1}^k G_i^{(t)}$ denote the clusters induced by the k centers C_t. At stage $t+1$, since we assign points to clusters that have the nearest squared euclidean distance, we minimize:

$$e_k(X; G^{(t+1)}) \leq e_k(X, G(C_t)).$$

Now, recall that the k-means cost function is equal to the weighted sum of the intra-cluster variances: $e_k(X; G^{(t+1)}) = \sum_{j=1}^k v(G_j^{(t+1)}, c_j)$. When we update the cluster centers to their centroids (that are the points that minimize the squared Euclidean distance in these groups), we have for each group $v(G_j^{(t+1)}, c(G_j^{(t+1)})) \leq v(G_j^{(t+1)}, c_j)$. Thus we deduce that:

$$e_k(X; C_{t+1}) \leq e_k(G^{(t+1)}; C_t) \leq e_k(X; C_t).$$

Since $e_k(X; C) \geq 0$ and that we can never repeat twice the same partitions among the $O(\binom{n}{k})$ potential partitions[7], we necessarily converge into a local minimum after a finite number of iterations. $\square$

Let us now present a few remarkable observations on the k-means clustering:

Observation 1

Although that Lloyd's heuristic perform remarkably well in practice, it has been shown that in the worst-case we can have an exponential number of iterations, even in the planar case $d = 2$ [88, 42]. In 1D, Lloyd's k-means can require $\Omega(n)$ iterations until convergence [42]. But recall that the 1D case can be solved polynomially using dynamic programming.

Observation 2

Even if the k-means cost function as a unique global minimum, there can be exponentially many partition solutions that yield this minimum value (one single min but many argmin): For example, let us consider the 4 vertices of a square. For $k = 2$, we have two optimal solutions (from the vertices of the parallel edges). Let us now clone $\frac{n}{4}$ copies of this square, each square located very far away to each other, and consider $k = \frac{n}{4}$: In that case, there exists 2^k optimal clusterings.

[7] The number of distinct partitions of a set of n elements into k non-empty subsets is defined by the *second kind of Stirling number*: $\left\{ n \atop k \right\} = \frac{1}{k!} \sum_{j=0}^k (-1)^{k-j} \binom{k}{j} j^n$.

Observation 3

In some cases, after assigning points to clusters in Lloyd's batched heuristic, we can obtain *empty clusters*: This case happens rarely in practice but its probability increases with the dimension. Thus one has to take care when implementing Lloyd's k-means of these potential empty cluster exceptions. One such configuration is illustrated in Figure 7.5. Note that this problem is in fact a blessing, since we can choose new center points in order to reinitialize those empty clusters while ensuring that the sum of cluster variances decreases.

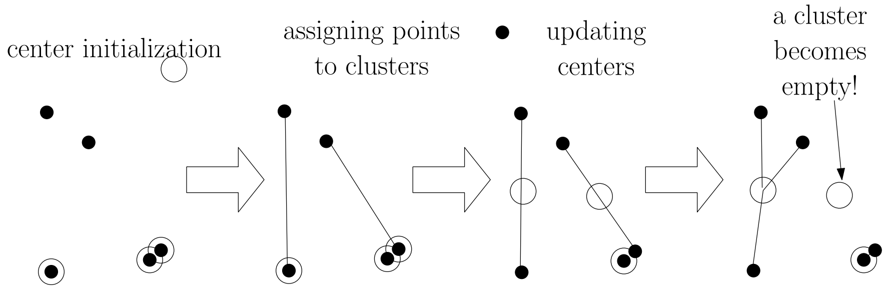


Figure 7.5 Empty cluster exceptions in Lloyd's heuristic: Cluster centers are depicted with large circles. Initialization followed by an assignment step with a center relocation step, and new assignment step. One of the cluster becomes empty.

Lloyd's k-means are a local heuristic that starts from a given initialization (either induced by an initial set of k prototypes, or by a given starting partition that induces the centroid prototypes) and guarantees monotonous convergence to a local minimum. We shall now describe a few initialization methods: That is, global heuristics for k-means.

7.4 Initializing k-means with global heuristics

7.4.1 Random seed initialization (as known as Forgy's intialization)

Let us choose the k distinct seeds randomly from X (for example, by sampling uniformly the k indices from $[n] = \{1, ..., n\}$). There are $\binom{n}{k}$ such different random drawings. Then we create the group partition $G(C) = \{G_1, ..., G_k\}$

from the seed variates $C = \{c_1, ..., c_k\}$. There is no theoretical guarantee that $e_k(X, G)$ is close to the global minimum $e_k^*(X, G) = \min_C e_k(X, G)$. Thus in order to increase the chance of finding a good initialization, not too far from $e_k^*(X, G)$, we can initialize l times to get the seed sets $C_1, ..., C_l$ and keep the best seeds that yielded the best k-means cost so far: That is, keep C_{l^*} with $l^* = \text{argmin}_l e_k(X; G(C_l))$. This method is sometimes called *Forgy's initialization* with l restarts in software packages.

7.4.2 Global k-means: The best greedy initialization

In the *global k-means*, we first choose randomly the first seed c_1, then greedily choose the seeds c_2 to c_k. Let $C_{\leq i} = \{c_1, ..., c_i\}$ denote the set of the first i seeds. We choose $c_i \in X$ in order to minimize $e_i(X, C_{\leq i})$ (there are only $n-i+1$ possible choices that can be exhaustively tested). Finally, we consider for c_1 all the n potential choices $c_1 = x_1$, ..., $c_1 = x_n$, and we keep the best seed set.

7.4.3 k-Means++: A simple probabilistically-guaranteed initialization

Let us now consider a probabilistic initialization that guarantees with high probability a good initialization. Denote by $e_k^*(X) = \min_C e_k(X; C) = e_k(X, C^*)$ the global minimum value of the k-means cost, with $C^* = \text{argmin}_C e_k(X; C)$. A $(1+\epsilon)$-approximation of the k-means is defined by a set of prototypes C such that:

$$e_k^*(X) \leq e_k(X, C) \leq (1+\epsilon) e_k^*(X).$$

In other words, the ratio $\frac{e_k(X,C)}{e_k^*(X)}$ is at most $1+\epsilon$.

The *k-means++ initialization* choose iteratively the seeds by weighting the elements x_i's according to the squared Euclidean distance of x_i to the already chosen seeds. Let $D^2(x, C)$ denote the minimum squared Euclidean distance of x to an element of C: $D^2(x, C) = \min_{c \in C} \|x - c\|^2$.

For a weighted set X, the k-means++ initialization procedure writes as follows:

– Choose c_1 uniformly randomly in X. If we had shuffled X beforehand, we set $C_{++} = \{c_1\}$.

– For $i = 2$ to k

Draw $c_i = x \in X$ with probability:

$$p(x) = \frac{w(x)D^2(x, C_{++})}{\sum_y w(y)D^2(y, C_{++})}$$

$C_{++} \leftarrow C_{++} \cup \{c_i\}$.

Theorem 6 (k-means++ [3])

k-Means++ probability initialization guarantee with high probability that $\mathbb{E}[e_k(X, C_{++})] \leq 8(2 + \ln k)e_k^*(X)$.

That is, the k-means++ are $\tilde{O}(\log k)$ competitive. The notation $\tilde{O}(\cdot)$ emphasizes the fact that the analysis is probabilistic in expectation. The technical proof is reported in the paper [3]. Here, to give a flavor of the tools used in the proof, we shall give an elementary proof in the case of $k = 1$ cluster. That is, we choose randomly a point $x_0 \in X$. Let c^* denote the center of mass of X.

We have:

$$\mathbb{E}[e_1(X)] = \frac{1}{|X|} \sum_{x_0 \in X} \sum_{x \in X} \|x - x_0\|^2.$$

We shall use the following *variance-bias decomposition* for any z:

$$\sum_{x \in X} \|x - z\|^2 - \sum_{x \in X} \|x - c^*\|^2 = |X| \|c^* - z\|^2.$$

Thus we deduce that

$$\begin{aligned}
\mathbb{E}[e_1(X)] &= \frac{1}{|X|} \sum_{x_0 \in X} \left(\sum_{x \in X} \|x - c^*\|^2 + |X| \|x_0 - c^*\|^2 \right), \\
&= 2 \sum_{x \in X} \|x - c^*\|^2, \\
&= 2e_1^*(X).
\end{aligned}$$

Hence, in the case of $k = 1$ cluster, by randomly choosing the first seed c_1 uniformly in X, we guarantee in expectation a 2-approximation.

7.5 Application of k-means to Vector Quantization (VQ)

7.5.1 Vector quantization

In *vector quantization* (VQ for short), we are given a set $X = \{x_1, ..., x_n\}$ that we seek to encode with words $c_1, ..., c_k$. This dictionary of words is called a *codebook*. We define the encoding and decoding functions as follows:

– quantization function $i(\cdot)$: $x \in \mathbb{R}^d \rightarrow \{1, ..., k\}$

– decoding function: $c(\cdot)$

To compress and code a message $(t_1, ..., t_m)$ of m elements of an alphabet X of n characters ($t_i \in X$), we associate to each t_i its code $i(t_i)$ via the encoding function $i(\cdot)$ that belongs to the k words of the code book. For example, we may quantize the 24-bit colors of an image (encoded using 3 color channels, 'R' for red, 'G' for green, and 'B' for blue) into k distinct color levels using vector quantization (the k-prototypes of the k-means ran on all the pixel colors). Thus instead of coding an image of dimension $m = w \times h$ pixels using $24m$ bits, we rather code the (R, G, B) colors of a *palette* of size k, and then encode the pixel colors using $m \times \log k$ bits (the contents of the image). Thus we save memory storage or shrink communication time for sending an image over a digital channel.

The distortion error introduced by the quantization of colors into an alphabet of k letters is $E = \frac{1}{n}\sum_{i=1}^{n} \|x - c(i(x))\|^2$, the *Mean Square Error* (*MSE*).

k-Means clustering allows one to find a code book minimizing (locally) the MSE: $e_k(X, C) = v(X) - v(C)$. Here, the variance denotes the information spread, and we seek to minimize this loss of information. Indeed, we can rewrite the k-means cost function as follows:

$$\begin{aligned} e_k(X, C) &= \sum_{i=1}^{n} \min_{j=1}^{k} \|x_i - c_j\|^2, \\ &= v(X) - v(C), \text{with } C = \{(n_j, c_j)\}_{j=1}^{k} \end{aligned}$$

Thus, k-means can be reinterpreted as the minimization of the difference between the variances of X and its quantization by k centers C. In other words, quantization asks to minimize the difference between the variance of a discrete random variable on n letters and the variance of a discrete random variable on k letters. The variance plays the role of information, and we seek to minimize this difference of information between these two random variables. In *Information Theory* (IT), this is related to rate distortion theory.

7.5.2 Lloyd's local minima and stable Voronoi partitions

For all $x_i \in X$, we associate a *label* in $\mathbb{N}$:

$$l_C(x) = \arg \min_{j \in \{1,\ldots,k\}} \|x - c_j\|^2$$

We can extend this labeling function to the full space $\mathbb{X}$, and we obtain a partition of $\mathbb{X}$ called a *Voronoi diagram*. This partition of space is illustrated in Figure 7.6. A *Voronoi cell* V_j of the Voronoi diagram is defined by:

$$V_j = \{x \in \mathbb{R}^d \ : \ \|x - c_j\| \leq \|x - c_l\| \ \forall l \in \{1, ..., n\}\}.$$

Here, let us remark that the squared Euclidean distance or the ordinary Euclidean distance yields to the same Voronoi diagram that decomposes the space into proximity cells. In fact, Voronoi cells do not change if we apply any strictly monotonous function on the base distance. The square function is such an example of a monotonous function on $\mathbb{R}_+$.

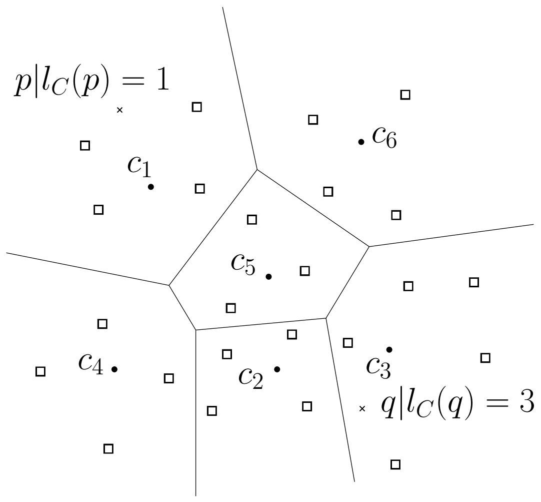


Figure 7.6 Voronoi diagram induced by k centers C, called generators, and Voronoi partition of X induced by C.

Now, let us note that once Lloyd's k-means heuristic has converged, the cluster groups G_i make a Voronoi partition, and have the *convex hulls* (co) of their groups pairwise disjoint: $\forall i \neq j, \quad \mathrm{co}(G_i) \cap \mathrm{co}(G_j) = \emptyset$ where:

$$\mathrm{co}(X) = \{x : x = \sum_{x_i \in X} \lambda_i x_i, \sum_{i=1}^{n} \lambda_i = 1, \lambda_i \geq 0\}.$$

7.6 A physical interpretation of *k*-means: The inertia decomposition

Let us consider $X = \{(x_i, w_i)\}_i$ as n body masses located at positions x_i's with respective weights w_i's. In Physics, the concept of *inertia* measures the resistance of a body when we move it around a given point. We measure the *total inertia* $I(X)$ of a point cloud X as the sum of squared Euclidean distances of points of X with respect to its center of mass $c = \sum_{i=1}^{k} w_i x_i$:

$$I(X) := \sum_{i=1}^{n} w_i \|x_i - c\|^2.$$

Thus, when we increase the masses on points, inertia increases too (*i.e.*, it becomes more difficult to rotate the point set around its center of mass). Similarly, when we consider points getting farther away of the center of mass, it also becomes harder to rotation this point cloud around its center of mass c. Therefore k-means can be reinterpreted physically as the task to identify k groups such that the sum of inertia of these groups with respect to their barycenters is minimal. *Huygens' formula* report an invariant or a mathematical identity between the total inertia of the system and its decomposition into the sum of the intra-group inertia plus the inertia of the inter-groups:

Theorem 7 (Huygens' formula: Inertia decomposition)

The total inertia $I(X) = \sum_{i=1}^{n} w_i \|x_i - \bar{x}\|^2$ equals to $I_{\text{intra}}(G) + I_{\text{inter}}(C)$ with the intra-group inertia $I_{\text{intra}}(G) = \sum_{i=1}^{k} I(G_i) = \sum_{i=1}^{k} \sum_{x_j \in \mathcal{G}_i} w_j \|x_j - c_i\|^2$ and the inter-group inertia $I_{\text{inter}}(C) = \sum_{i=1}^{k} W_i \|c_i - c\|^2$ (a unique centroid c) with $W_i = \sum_{x \in G_i} w(x)$.

Figure 7.7 illustrates two decompositions of inertia that have the same total inertia. Since the total inertia is invariant, minimizing the intra-group inertia amounts to maximize the inter-group inertia.

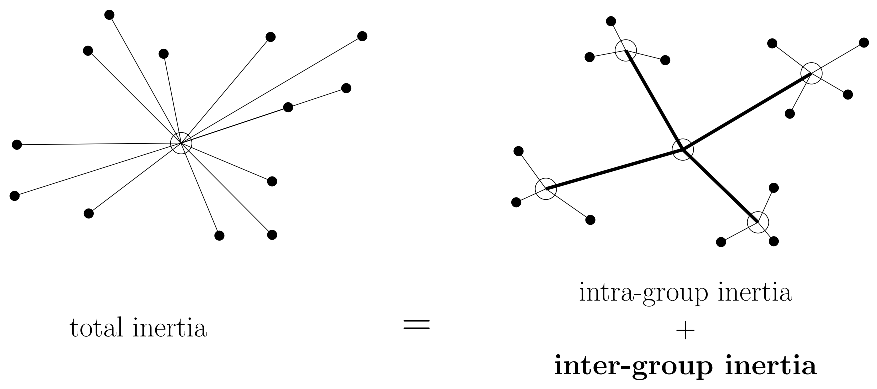


Figure 7.7 The total inertia of a system of point masses is invariant by group decomposition. k-Means optimizes the decomposition that minimizes the intra-group inertia.

7.7 Choosing k in k-means: Model selection

Until so far, we have considered that the number of clusters k was prescribed, and known beforehand. This is not true in practice when one performs exploratory data analysis, and k has to be guessed as well. Finding the right value of k is an important problem referred to as *model selection* in the literature. For any value of k, we can consider the optimal k-means cost function $e_k^*(X)$ (that can be estimated empirically in practice, say using Lloyd's heuristics on several initializations). Let us notice that $e_k(X)$ decreases monotonically until reaching $e_n(X) = 0$ (in that case, each point is trivially allocated to its own cluster).

7.7.1 Model selection via the elbow method

To choose a correct value for k, one can use the so-called *elbow method*. This is a visual procedure: First, we plot the function $(k, e_k(X))$ for $k \in [n] = \{1, ..., n\}$, and we choose k that defines the inflexion point: the elbow (separating the forearm from the arm). The reason to choose this value of k is that for small k values, the sum of cluster variances decreases quickly and then starting from some value, the sum of variances describes *a plateau*. This visual inspection method is called the "elbow method" because the function $f(k) = e_k(X)$ looks like an arm on practical data-sets (with the plateau being the forearm): The elbow returns the optimal number of clusters. One drawback of this method is that it is computationally very expensive, and sometimes (depending on the data-sets), the inflexion point between the sharp decrease and the plateau is not so well defined!

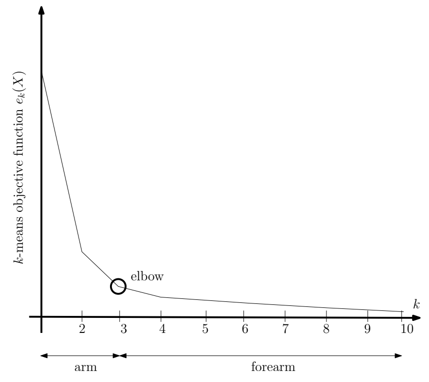


Figure 7.8 Choosing k with the elbow method: the elbow defines the value of k that separates the area of high decrease (the arm) to the plateau area (the forearm).

7.7.2 Model selection: Explaining variance reduction with k

We calculate the *proportion of variance* that is explained by k classes:

$$R^2(k) = \frac{I_{\text{inter}}(k)}{I_{\text{total}}}.$$

We have $0 < R^2(k) \leq 1$. We then choose k^* that minimizes the ratio $\frac{R^2(k)}{R^2(k+1)}$:

$$k^* = \arg\min_k \left(\frac{R^2(k)}{R^2(k+1)} \right).$$

We have presented two essential methods to choose the right value of k, the number of clusters. In machine learning, k denotes the *complexity of the model*, and choosing k is therefore called the *model selection* problem. There exists some algorithms that perform clustering without requiring to know beforehand k. For example, *affinity propagation* [32] (2007) is such a popular algorithm, or a fast convex relaxation minimization of k-means [59] (2007).

7.8 Parallel *k*-means clustering on a cluster of machines

There are many ways to design a parallel version of Lloyd's k-means heuristic on a cluster of computers : That is, on a set of interconnected machines that we consider as a "super-computer" with distributed memory (one local memory for each machine). As usual, for sake of simplicity, we consider that each computer has a single processing core (a processing unit) and associates a node to each machine to describe the communication links by a graph. Those different processors communicate with each other by sending and receiving messages using the MPI interface: The *Message Passing Interface*. Sending a message has a communication cost that is split into two terms: one for the latency of initialization the communication and one that is proportional to the length of the message. Sending structured data requires first to *serialize* it into a universal "string" before sending (encoding) at the sender node and to reconstruct it (de-serialize the string to reconstruct the structure) at the receiver node.

A k-means clustering problem can be characterized by the number of attributes (that is, the dimension of the data point cloud), the number of data elements n, and the number of clusters, k. We consider $k << n$ (that is, $k = o(n)$) so that all cluster centers requiring a memory space $O(dk)$ can be stored into the local memory of each machine. Since in practice the memory (RAM) is fixed (that is, $O(1)$), that means that we consider theoretically the special case of $k = O(1)$ here. However, the number of elements n is considered (very) large compared to k, so that the full data-set X need to be distributed among the P processors (since X cannot be fully contained in the RAM of a single computer).

In order to design a simple but efficient parallel k-means heuristic, we rely on the following composability/decomposability theorem:

Theorem 8 (Composability of barycenters)

Let X_1 and X_2 be two weighted data-sets with respective overall weights $W_1 > 0$ and $W_2 > 0$. We have:

$$\boxed{\bar{x}(X_1 \cup X_2) = \frac{W_1}{W_1 + W_2}\bar{x}(X_1) + \frac{W_2}{W_1 + W_2}\bar{x}(X_2)}$$

where $\bar{x}(X_i)$ denotes the barycenter for X_i, for $i \in \{1, 2\}$.

This property turns out to be essential for distributed the centroid computation on a partition of X into P subsets $X_1, ..., X_P$ where P is the number of processors (or machines of the cluster). The parallel algorithm is described in pseudo-code in Algorithm 5.

At run time, each processor knows the overall number of processors of the cluster, P, by using the function `MPI_Comm_size()`, and its rank number indexed between 0 and $P-1$ by using the standard MPI function `MPI_Comm_rank()`. It is the task of processor P_0 (the root machine) to initialize the k cluster prototypes and to broadcast them to all other processors using the primitive `MPI_Bcast`(C, root processor). Then we loop until convergence using a `while` structure: Each processor P_l computes the labeling of its group of data X_l, and the cumulative sum of the vectors of the k groups corresponding to P_l with the local cardinality of each cluster. Then we aggregate and broadcast all those group cardinals using the MPI primitive `MPI_Allreduce`. The aggregation operation (that is associative and commutative) can be chosen among a set of binary operators like + or min, etc. We specify this binary operation as an argument of the MPI primitive `MPI_Allreduce`: Here, it is `MPI_SUM` to indicate the cumulative sum operation. Since some clusters in some partial data-sets associated to machines may be empty, we take of those cases when computing the local centroids (see the $\max(n_j, 1)$ operation in Algorithm 5).

A complete source code using the C API of OpenMPI is syntaxically different from the algorithm 5 written in the pseudo-code as the arguments in the MPI primitives ask for the length of the message, the type of data to be communicated, etc.

```
/* Distributed k-means clustering in MPI */
$p$ = MPI_Comm_size();
$r$ = MPI_Comm_rank();
previousMSE = 0;
/* Mean Square Error, the cost function for the k-means */
MSE = $\infty$;
if $r = 0$ then
    /* Initialize randomly the cluster seeds */
    Initialize $C = (c_1, ..., c_k)$;
    MPI_Bcast($C, 0$);
end
while MSE $\neq$ previousMSE do
    previousMSE = MSE;
    MSE$'$ = 0;
    for $j = 1$ to $k$ do
        $m'_j = 0$;
        $n'_j = 0$;
    end
    for $i = r(n/p)$ to $(r+1)(n/p) - 1$ do
        for $j = 1$ to $k$ do
            Calculate $d_{i,j} = d^2(x_i, m_j) = \|x_i - m_j\|^2$;
        end
        Find the closest centroid $m_l$ to $x_i$: $l = \arg\min_j d_{i,j}$;
        /* Update stage */
        $m'_l = m'_l + x_i$;
        $n'_l = n'_l + 1$;
        MSE$'$ = MSE$'$ + $d^2(x_i, m_l)$;
    end
    /* Aggregate: make use of the composability property of
       centroids */
    for $j = 1$ to $k$ do
        MPI_Allreduce($n'_j, n_j$, MPI_SUM);
        MPI_Allreduce($m'_j, m_j$, MPI_SUM);
        /* To prevent dividing by zero */
        $n_j = \max(n_j, 1)$;
        $m_j = m_j / n_j$;
    end
    /* Update the cost function */
    MPI_Allreduce(MSE$'$, MSE, MPI_SUM);
end
```

Algorithm 5: Lloyd's parallel k-means heuristic using MPI.

In this distributed implementation of k-means, we optimize the sequential code by an optimal speed-up P.

7.9 Evaluating clustering partitions

In order to evaluate the performances of the various clustering techniques (like the various k-means local/global heuristics), it is important to have *ground-truth data-sets* that tell us for each data element is true cluster membership. Without these ground-truth data sets, we could only perform a subjective or qualitative evaluation of the clustering methods. Although that in 2D, Human eyes can amazingly evaluate whether the obtained clustering is good or not, it becomes impossible to visualize in dimensions $d > 3$.

When ground-truth data-sets are available (say, data-sets annotated by experts), we can compute various metrics that are quantitative values that measure the *similarity of two partitions*: the one induced by the labels in the ground-truth data-set (assumed to be the optimal clustering by definition!) and the one reported by an automatic clustering algorithm.

7.9.1 The Rand index

The *Rand index* (1971) computes the similarity of two partitions as follows: Let $G = \uplus G_i$ and $G' = \uplus G'_i$ be the cluster decomposition of the k-means heuristic and of the ground-truth data-set, respectively.

We compare all the $\binom{n}{2}$ pairs (x_i, x_j) of points, and count those that belong to the same cluster (a) from those that are found to belong to different clusters (b). Thus we obtain the Rand index that belongs to the interval $[0, 1]$:

$$\boxed{\mathrm{Rand}(G, G') = \frac{a+b}{\binom{n}{2}}}$$

with

– a: $\#\{(i,j) \ : \ l(x_i) = l(x_j) \ \wedge \ l'(x_i) = l'(x_j)\}$,

– b: $\#\{(i,j) \ : \ l(x_i) \neq l(x_j) \ \wedge \ l'(x_i) \neq l'(x_j)\}$,

where $l(\mathrm{cot})$ and $l'(\mathrm{cot})$ are the two cluster labeling functions of the ground-truth clustering and the automatic clustering.

Notation: $\mathrm{condition}_1 \wedge \mathrm{condition}_2$ denotes that both conditions have to be true in order to be true (logic AND operator). Let us remark that the

Rand index avoids us to relabel the k groups in order to make the two partitions compatible with each other: Indeed, there are $k!$ such permutation relabeling that we should take into account otherwise, and it is therefore not computationally-tractable (since $k!$ grows exponentially with k) to consider them in practice! A more sophisticated implementation of the Rand index often used in practice is called the *adjusted Rand index* [49] (1985).

7.9.2 Normalized Mutual Information (NMI)

The *Normalized Mutual Information* (*NMI*) is a notion that is well-defined in Information Theory. Let $n_{j,j'} = \{x \in G_j \ \wedge \ x \in G'_{j'}\}$. Then the NMI is defined as:

$$\boxed{\mathrm{NMI}(G, G') = \frac{\sum_j^k \sum_{j'}^{k'} n_{j,j'} \log \frac{n \times n_{j,j'}}{n_j n_{j'}}}{\sqrt{\left(\sum_j^k n_j \log \frac{n_j}{n}\right)\left(\sum_{j'}^{k'} n'_j \log \frac{n'_j}{n}\right)}}}$$

NMI is an estimation of the information-theoretic quantity:

$$\frac{I(X;Y)}{\sqrt{H(X)H(Y)}},$$

where $I(X;Y)$ denotes the *mutual information* between two random variables, and $H(\cdot)$ denotes the *Shannon entropy* of a random variable.

7.10 Notes and references

We described a way to cluster data by minimizing a cost function: the sum of intra-cluster variances of the k clusters. Historically, the methodology of the k-means has been first introduced by Hugo Steinhaus [81] in 1956 (by studying the inertia of a body), and has been rediscovered many times independently later on (like in vector quantization, VQ), etc. In this chapter, we have described the usual k-means techniques. A full description of k-means requires its own textbook! Depending on the cost function, one can obtain more or less efficient optimization algorithms, and the obtained clustering can be more or less adapted to the data-sets. In fact, when one describes axiomatically the properties of a good clustering, it can be shown that there *does not* exist any cost function to optimize that fulfills those properties [55] (see also [91, 17]).

Lloyd's batched heuristic has been first reported in details in [62], in 1957. In practice, Hartigan's single-swap heuristic (described in the exercise 7.12) is getting again more and more popular since it is more efficient and provably reaches better local minima than Lloyd's method. The k-means++ probabilistic initialization dates back from 2007. A deterministic initialization for k-means is described in [65] (2000): One get a $(1 + \epsilon)$-approximation in time $O(\epsilon^{-2k^2 d} n \log^k n)$. Among the NP-hard problems, k-means is rather "easy" to approximate because it admits a *Polynomial Time Approximation Scheme* (or *PTAS* for short) [5]: That is, for any $\epsilon > 0$, one gets a $(1+\epsilon)$-approximation of the k-means cost function in polynomial time.

k-Means have been reported on a distributed memory parallel architecture using the MPI interface in [26]. Lloyd's heuristic is updating the point assignment to clusters at each stage (batched k-means): one can also update after each point relocation by considering the points one by one. This is precisely the MacQueen's heuristic [63] (1967) that is mentioned in exercise 7.12. k-Means++ is intrinsically a sequential algorithm and its generalization to parallel architectures, termed k-means$||$, has been proposed in [10]. Nowadays, with the wide availability of big size data-sets, one even seeks to cluster large data-sets with billions (k) of clusters [8]. The *core-sets* approximation techniques [28] allows one to reduce very large data-sets into tiny data-sets by trading the exact cost function minimization by a controlled approximation of it. Another construction method of such core-sets in parallel has been reported in[11] (2013). Another hot topic in data clustering is to be able to cluster data among different entities while preserving the privacy of data elements. In [87], a method is given to perform privacy-preserving data clustering on *vertically partitioned data*[8] with Lloyd's k-means heuristic.

We have emphasized on the fact that the k-means objective function seek to detect globular-shaped clusters. Although the k-means is often used in practice, it is not (by far!) a universal solution for clustering. For example, Figure 7.9 depicts a data-set that is easy to cluster by Human eye-brain systems; However, the k-means optimization will not obtain a desired partitioning. This is because k-means report Voronoi partitions However, this kind of data-set is handled using kernel[9] k-means [25].

[8] Vertical partitioning means that each entity has only a block of attributes

[9] It is mathematically always possible to separate data by increasing by lifting the features into higher dimensions using a *kernel mapping*.

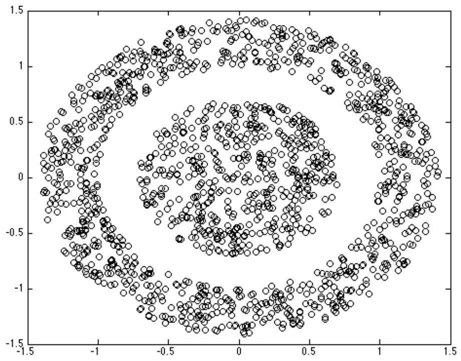


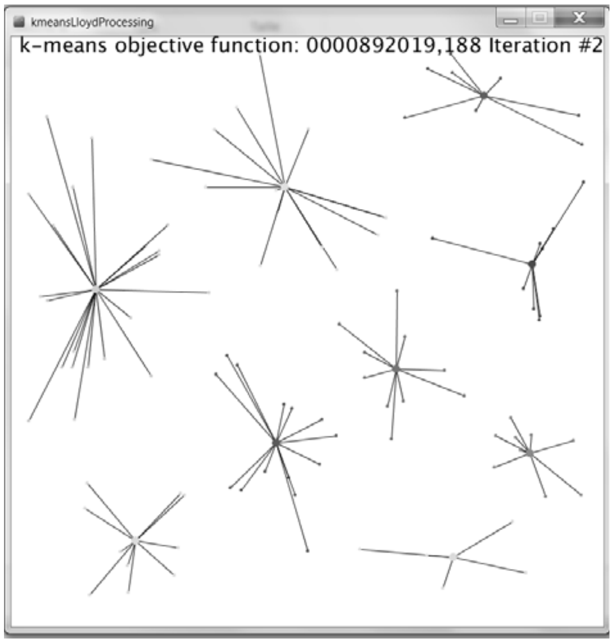
Figure 7.9 Example of a data-set for which k-means fails to cluster appropriately into two clusters. Indeed, k-means is limited to report Voronoi partitions with disjoint group convex hull. In practice, such a data-set is clustered using a kernel method.

7.11 Summary

Exploratory data analysis consists in finding structures in data-sets that bring knowledge of these data-sets. Clustering is a set of techniques that partitions data into homogeneous groups, and thus allows one to discover classes, with potential semantic meaning for each class. Clustering by k-means asks to minimize the weighted sum of intra-cluster variances by assigning to each group a center: its prototype that plays the role of the model for that cluster. That is, k-means clustering belongs to the family of *model-based clustering*. Minimizing the k-means objective function is a NP-hard problem in general, and the celebrated Lloyd's heuristic consists in repeating until convergence the following two steps: (1) assignment of data points to their closest cluster center, and (2) update the cluster centers by setting the centers (prototypes) to their cluster centroids. Lloyd's heuristic is guaranteed to converge monotonically to a local minimum that is characterized by a Voronoi partition induced by the cluster centers. Since we do not know *a priori* the number of clusters k, we need to estimate it by performing a *model selection*: a usual rule of thumb consists in choosing the value of k that minimizes the ratio of the sum of intra-cluster variances for k and $k+1$, and that can be visually interpreted as the elbow (hence the name, elbow method), the inflexion point in the graph plot of the cost function $e_k(\cdot)$. Lloyd's k-means can be easily parallelized on a distributed memory architecture by using the MPI interface and by using the *decomposition property of centroids*: The centroid (or barycenter) of a data set partitioned into groups amounts to equivalently compute the barycenter of the centroids (or barycenters) of the groups.

Processing code for Lloyd's *k*-means algorithm

Figure 7.10 displays a snapshot of the `processing.org` program.



k−means objective function: 0000892019,188 Iteration #2

Figure 7.10 Snapshot of the `processing` code for Lloyd's k-means heuristic.

WWW source code: `kmeansLloydProcessing.pde`

7.12 Exercises

Exercise 1: *Barycenter and variance with non-normalized positive weights*

– For a positive vector $w = (w_1, ..., w_n) \in \mathbb{R}^d_+$ (not normalized to one) on data-set $X = \{x_1, ..., x_n\}$, prove that weighted sum of squared Euclidean distance to the center $\sum_{i=1}^n w_i \|x_i - c\|^2$ is minimized for the barycenter $\bar{x}$:

$$\bar{x} = \sum_{i=1}^n \frac{w_i}{W} x_i,$$

where $W = \sum_{i=1}^n w_i$ is the total sum of weights and that the non-normalized variance can be written as:

$$v(X, w) = \sum_{i=1}^n w_i \|x_i - \bar{x}\|^2 = \sum_{i=1}^n w_i \|x_i\|^2 - W\bar{x}^2.$$

– Observe that this formula amounts to take normalized weights $\tilde{w}_i = \frac{w_i}{W}$ in the classic (normalized weight) formula.

– What happens when some weights are negative? Can we still guarantee the uniqueness of the minimizer?

– Deduce the composability formula of barycenters: Let $\{X_i\}_{i \in \{1,...,k\}}$ be k weighted data-sets with respective total weights W_i. Prove that:

$$\bar{x}(\uplus_{i=1}^k X_i) = \sum_{i=1}^k \frac{W_i}{\sum_{j=1}^k W_j} \bar{x}(X_i),$$

where $\bar{x}(X_i)$ are the barycenters of the X_i's.

Exercise 2: *Center of mass for scalars ($d = 1$)*

Prove that the *arithmetic mean* $c = \frac{1}{n}\sum_{i=1}^n x_i$ is also an equilibrium centers since we have the following property:

$$\sum_{x_i < c} (c - x_i) = \sum_{x_i \geq c} (x_i - c).$$

Exercise 3: *Bias-variance decomposition*

Let $v(X, z) = \sum_{x \in X} \|x - z\|^2$ and $v(X) = v(X, \bar{x})$ with $\bar{x} = \frac{1}{n}\sum_i x_i$.

– Prove that $v(X, z) = v(X) + n\|\bar{x} - z\|^2$. Deduce that the center of mass $\bar{x}$ minimizes $v(X, z)$.

– Generalize this decomposition to weighted point sets $X = \{(x_i, w_i)\}_i$.

– Interpret X as a discrete random variable, and prove the bias-variance decomposition formula for an arbitrary random variable.

Exercise 4: *k-Medoids (as known as* discrete k-means*)*

Let us minimize the k-means cost function by constraining the prototypes c_j to belong to the data-elements, the x_i's.

- Prove using the bias-variance decomposition that the best cost of the k-*medoids* is at most twice the best cost of k-means.
- Deduce a heuristic for the k-means where prototype are constrained to belong to the initial data-set (batched assignment).
- Provide an upper bound on the maximal number of iterations of your heuristic.

Exercise 5: *Inta-cluster distance minimization and inter-cluster distance*

maximization

Let $X = \{x_1, ..., x_n\}$ be a data-set of n elements (quantitative or categorical attributes), and $D(x_i, x_j) \geq 0$ a dissimilarity function between any two arbitrary elements $x_i \in X$ and $x_j \in X$. Prove that for a given partition of X into k clusters $C_1, ..., C_k$ that minimizing the intra-cluster pairwise distances $\sum_{l=1}^k \sum_{x_i \in C_l} \sum_{x_j \in C_l} D(x_i, x_j)$ is equivalent to maximize the pairwise inter-cluster distances $\sum_{l=1}^k \sum_{x_i \in C_l} \sum_{x_j \notin C_l} D(x_i, x_j)$. For categorical attribute data-sets, one can consider the *Jaccard distance*, $D(x_i, x_j) = \frac{|x_i \cap x_j|}{x_i \cup x_j}$, and cluster with the k-medoid technique described in the exercise 7.12.

Exercise 6: *MacQueen's local k-means heuristic [63]*

MacQueen's local k-means heuristic updates iteratively the cluster prototypes by assigning the points one by one to clusters until it converges:

- Initialize $c_j = x_j$ for $j = 1, ..., k$
- Assign incrementally data elements $x_1, ..., x_n$ in a cyclic sequence until convergence: Assign $x_i$to its nearest cluster center c_j of C, and update this center: we remove x_i from its current center and assign it to its new center.

– Prove the following update formula:

$$\begin{aligned}
c_{l(x_i)} &\leftarrow \frac{n_{l(x_i)} c_{l(x_i)} - x_i}{n_{l(x_i)} - 1}, \quad n_{l(x_i)} \leftarrow n_{l(x_i)} - 1 \\
l(x_i) &= \mathrm{argmin}_j \|x_i - c_j\|^2 \\
c_{l(x_i)} &\leftarrow \frac{n_{l(x_i)} c_{l(x_i)} + x_i}{n_{l(x_i)} - 1}, \quad n_{l(x_i)} \leftarrow n_{l(x_i)} + 1
\end{aligned}$$

– Prove that local minima match the local minima of Lloyd's batched k-means.

– What is the complexity of MacQueen's heuristic?

Exercise 7: Hartigan's k-means heuristic*: Swap a point from a cluster to another cluster [84]*

We propose the following iterative k-means local heuristic: Consider in a cyclic order the elements x_i, one by one. For a given x_i currently belonging to cluster $G_{l(x_i)}$ with $l(x_i) = \mathrm{argmin}_{j \in \{1,\dots,k\}} \|x_i - c_j\|^2$ its membership, we move x_i into another cluster G_l iff. the k-means cost function decreases.

1. Write mathematically $\Delta(x_i, l)$: the gain of the cost function when x_i is swapped from $G_{l(x_i)}$ to G_l. For x_i being swapped from a source cluster $G_s$to a target cluster G_t, prove the following formula:

$$\Delta(x_i; s \to t) = \frac{n_t}{n_t + 1}\|c_t - x_i\|^2 - \frac{n_s}{n_s - 1}\|c_s - x_i\|^2$$

2. Prove that Hartigan's local minima are a proper subset of Lloyd's local minima.
3. Give in pseudo-code the Hartigan's k-means local heuristic. What is the complexity of this algorithm?
4. Observe that this heuristic always guarantees non-empty clusters at any time (and behaves differently from Lloyd's heuristic that may procedure empty cluster exceptions).

Exercise 8: *Horizontally versus vertically separated k-means parallel clustering [87]*

Consider clustering n data of d attributes using a cluster of P machines. Assume that $d \gg n$ and propose a parallel implementation of the k-means algorithm by distributing the features among the machines (thus a portion of each datum is stored on each machine). Compare your *vertically separated* implementation of k-means with the *horizontally separated* parallel k-means (when data elements are partitioned and distributed among the machines).

Exercise 9: *** k-Means clustering with Bregman divergences [12]*

The k-means cost function can be generalized to *Bregman divergences* as follows: $e_k(X,G) = \sum_{i=1}^n \min_{j\in\{1,\ldots,k\}} D_F(x_i,c_j)$. Bregman divergences are defined for a strictly convex and differentiable generator $F(x)$ by:

$$D_F(x,y) = F(x) - F(y) - (x-y)^\top \nabla F(y),$$

where $\nabla F(y) = (\frac{\mathrm{d}}{\mathrm{d}y^1}F(y), ..., \frac{\mathrm{d}}{\mathrm{d}y^d}F(y))$ denotes the gradient operator (vector of partial derivatives).

1. Prove that the squared Euclidean distance is a Bregman divergence but not the Euclidean distance.
2. Prove that the minimizer of $\min_c \sum_{i=1}^n w_i D_F(x_i,c)$ is the barycenter $\bar{x} = \sum_{i=1}^n w_i x_i$. (In fact, one can prove that the only distortion measures that ensure barycenters as minimizers are the Bregman divergences.)
3. Deduce the corresponding Bregman batches k-means and Bregman Hartigan's k-means heuristics.
4. Prove that the composability property of barycenters still holds for Bregman divergences.

Exercise 10: ** k-Modes [48]*

To cluster categorical data (that is, non-numerical data), one can use the *Hamming distance* between any two d-dimensional attribute vectors x and y: $D_H(x,y) = \sum_{j=1}^d 1_{x^j \neq y^j}$ where $1_{a\neq b} = 1$ iff. $a \neq b$ and zero otherwise. Hamming distance is a metric satisfying the triangular inequality. Let $t_{l,m}$ be the m-th category of the l-th dimension of an element.

1. Prove that the mode $m = (m^1, ..., m^d)$ with $m^j = t_{j,m^*}$ and $m^* = \mathrm{argmax}_m \#\{x_i^j = t_{j,m}\}$ maximizes $\sum_{i=1}^n w_i D_H(x_i, m)$ where $\#\{\cdot\}$ denotes the cardinality of a data-set. That is, in other words, for each dimension, we choose the dominant category for the mode.
2. Prove that the barycenter may not be necessarily unique by reporting a counter example.

3. Design a *k-mode clustering heuristic* that is inspired from k-means heuristics, and show how one can use it to cluster a collection of text documents.
4. Show how to combine k-means and k-modes to cluster mixed attribute vectors (with some dimensions being numerical, and some other dimensions categorical).

Exercise 11: *Hegselmann-Krause model for* opinion dynamics *[44]*

Consider a set of n individuals $p_1, ..., p_n$, represented as points of the d-dimensional space $\mathbb{R}^d$. At a given iteration, each individual updates its position as the center of mass of all individuals falling within a distance less than a prescribed threshold r (say, radius $r = 1$; thus including itself too), and repeat this process until convergence (when individuals do not move anymore). At convergence, there are at most $k \leq n$ distinct individuals (distinct opinions). Implement in MPI this algorithm. What is the complexity of your algorithm? How does this algorithm differ from the Lloyd's k-means algorithm?

Exercise 12: *** Barycenters for an arbitrary convex distance*

Let $D(\cdot,\cdot)$ be a strictly convex and twice differentiable distance function (not necessarily symmetric nor satisfying the triangular inequality). We define the barycenter $\bar{x}$ of a weighted point cloud $X = \{(x_i, w_i)\}_i$ as the minimizer of $\bar{x} = \arg\min_c \sum_{i=1}^n w_i D(x_i, c)$.

– Prove that this barycenter is unique.

– Provide a geometric interpretation when zeroing the gradient $\nabla_c(\sum_{i=1}^n w_i D(x_i, c))$: The barycenter is the unique point that cancels the vector field $V(x) = \sum_{i=1}^n w_i \nabla_x D(x_i, x)$.

Exercise 13: *** 1D k-means using dynamic programming [73]*

Although that k-means in a NP-hard problem, in 1D, one can get a polynomial-time algorithm using dynamic programming. First, one starts by sorting the n scalars of $X = \{x_1, ..., x_n\}$ in increasing order, in $O(n \log n)$ time. Therefore we assume that $x_1 \leq ... \leq x_n$.

– We seek for a relationship between the optimal clustering for k clusters from the optimal clustering for $k-1$ clusters. Let $X_{i,j} = \{x_i, ..., x_j\}$ denote the sub-set of scalars $x_i, ..., x_j$. Write the mathematical recurrence equation of the best clustering $e_k(X_{1,n})$ using the terms $e_{k-1}(X_{1,j-1})$ and $e_1(X_{j,n})$.

– Show how to find the optimal partition from the dynamic programming table using backtracking. What is the complexity of your algorithm?

– By preprocessing the n scalars into three cumulative sums $\sum_{l=1}^{j} w_l$, $\sum_{l=1}^{j} w_l x_i$ and $\sum_{l=1}^{j} w_l x_i^2$, show how to calculate $v(X_{i,j}) = \sum_{l=i}^{j} w_i \|x_l - \bar{x}_{i,j}\|$ in constant time, where $\bar{x}_{i,j} = \frac{1}{\sum_{l=i}^{j} w_l} \sum_{l=i}^{j} w_l x_l$. Deduce that an optimal k-means partitioning can be exactly calculated in 1D in $O(n^2 k)$ time.

8
Hierarchical clustering

A concise summary is provided at the end of this chapter, in §8.7.

8.1 Agglomerative versus divisive hierarchical clustering, and dendrogram representations

Hierarchical clustering is yet another technique for performing data exploratory analysis. It is an unsupervised technique. In the former clustering chapter, we have described at length a technique to partition a data-set $X = \{x_1, ..., x_n\}$ into a collection of groups called clusters $X = \uplus_{i=1}^{k} G_i$ by minimizing the k-means objective function (*i.e.*, the weighted sum of cluster intra-variances): In that case, we dealt with flat clustering that delivers a non-hierarchical partition structure of the data-set. To contrast with this flat clustering technique, we cover in this chapter another widely used clustering technique: Namely, *hierarchical clustering*.

Hierarchical clustering consists in building a binary merge tree, starting from the data elements stored at the leaves (interpreted as singleton sets) and proceed by merging two by two the “closest” sub-sets (stored at nodes) until we reach the root of the tree that contains all the elements of X. We denote by $\Delta(X_i, X_j)$ the distance between any two sub-sets of X, called the *linkage distance*. This technique is also called *agglomerative hierarchical clustering* since we start from the leaves storing singletons (the x_i's) and merge iteratively