1 Introduction and motivation

For a given document collection $\mathcal{D} = \{D_1, D_2, \ldots, D_m\}$ and a query Q one often is concerned with the following basic problems:

1. Find documents in $\mathcal{D}$ "related" to the query. If, for example, a "distance" between two documents D_i and D_j is given by the function $d(D_i, D_j)$ and a threshold $\texttt{tol} > 0$ is specified one may be interested in identifying the document subset $\mathcal{D}_{\texttt{tol}} \subseteq \mathcal{D}$ defined by

$$\mathcal{D}_{\texttt{tol}} = \{D \,:\, D \in \mathcal{D},\ d(Q, D) < \texttt{tol}\}. \tag{1.0.1}$$

2. Partition the collection $\mathcal{D}$ into disjoint subcollections $\pi_1, \pi_2, \ldots, \pi_k$ (called clusters) so that the documents in a cluster are more similar to each other than to documents in other clusters. The number of clusters k also has to be determined.

When "tight" clusters π_i, $i = 1, \ldots, k$ are available "representatives" $\mathbf{c}_i$ of the clusters can be used instead of documents to identify $\mathcal{D}_{\texttt{tol}}$. The substitution of documents by representatives reduces the data set size and speeds up the search at the expense of accuracy. The "tighter" the clusters are the less accuracy is expected to be lost. Building "high quality" clusters is, therefore, of paramount importance to the first problem. Applications of clustering to IR are in particular motivated by *the Cluster Hypothesis* which states that "closely associated documents tend to be related to the same requests."

The World Wide Web provides a huge reservoir of information and the result of a query to a search engine can return thousand of pages. Clustering can be used to partition these results into groups each of which captures a particular aspect of the query. So that, for example, when the query is "virus" results related to "computer virus" will be placed in one cluster, while the results related to "viruses that cause colds" will be located in another one. A number of Internet search engines cluster search results thus providing a user with an efficient navigation tool.[1] Natural steps to approach the two above-mentioned problems are:

Step 1. Embed the documents and the query into a metric space.

Step 2. Handle problems 1 and 2 above as problems concerning points in the metric space.

This book discusses in detail a particular family of algorithms that clusters a finite data set $\mathcal{A}$ in a finite-dimensional Euclidean space. The problem first presented as a discrete optimization problem of finding the "optimal" k-cluster partition of the data set (as this is traditionally done in clustering literature). Next we state the clustering problem as a continuous optimization problem to which various optimization techniques are applied. The book also discusses different choices of "distance-like" functions with a particular emphasis on Bregman and Csiszar divergences that already found many useful applications in optimization and have been recently introduced in machine learning and clustering literature.

Document collections are often changing with time (new documents may be added to the existing collection and old documents may be discarded). It is, therefore, of interest to address the clustering problem under the assumption $\mathcal{D} = \mathcal{D}(t)$ (i.e., the document collection $\mathcal{D}$ is time-dependent).

[1] See, for example, http://www.groxis.com,http://www.iboogie.tv,http://www.kartoo.com, http://www.mooter.com/, and http://vivisimo.com

1.1. A way to embed ASCII documents into a finite-dimensional Euclidean space

A vector space model maps documents into vectors in a finite-dimensional Euclidean space. A brief description of perhaps the simplest vector space model is the following. First, a sorted list of words that occur in all the documents is built, this list serves as a dictionary. Words that belong to a stop list[2] are removed from the dictionary. If the number of distinct words in the dictionary is n, then a vector $\mathbf{a}_i \in \mathbf{R}^n$ is built for document D_i, $i = 1, \ldots, m$ as follows: the first coordinate $\mathbf{a}_i[1]$ of $\mathbf{a}_i$ is the number of times the first word of the dictionary occurs in D_i (if the word does not occur in D_i, then $\mathbf{a}_i[1] = 0$). The second coordinate $\mathbf{a}_i[2]$ is the number of times the second word of the dictionary occurs in D_i, and so on.

We illustrate the construction by a simple example. Consider the following collection[3]:

$D_1 =$ We expect a lot from our search engines.

$D_2 =$ We ask them vague questions about topics that we're unfamiliar with ourselves and in turn anticipate a concise, organize response.

$D_3 =$ We type in principal when we meant principle.

After the stop words removal the sorted dictionary contains 17 words:

anticipate, concise, engines, expect, lot, meant, organize, principal,
principle, questions, response, search, topics, turn, type, unfamiliar, vague.

[2] A stop list is a list of words that are believed to have a little or no value as search or discrimination terms. For example, the stop list from the SMART system at Cornell University can be found at ftp://ftp.cs.cornell.edu/pub/smart/english.stop

[3] The text is borrowed from [16]

The vector space dimension $n = 17$, and we will be building vectors in $\mathbf{R}^{17}$. For example, the vector $\mathbf{a}_1$ corresponding to the document D_1 is given by

$$\mathbf{a}_1 = (0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0)^T.$$

The simple example indicates that one can expect sparse high-dimensional vectors (this is indeed the case in IR applications). A "term by document" matrix is the $n \times m$ matrix with columns being the vectors $\mathbf{a}_i$, $i = 1, \ldots, m$. Each column of the matrix $A = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$ represents a document and each row corresponds to a unique term in the collection. While this book is mainly concerned with document clustering, one can consider word/term clustering by focusing on rows of the "term by document" matrix. In fact simultaneous co-clustering of columns and rows may benefit separate clustering of columns only or rows only.

It is not uncommon that coordinates of $\mathbf{a}_i$ are weighted frequencies (and not just raw counts of word occurrences). Vectors $\mathbf{a}_i$ are usually L_2 normalized so that $\|\mathbf{a}_i\|_2 = (\sum \mathbf{a}_i^2[j])^{\frac{1}{2}} = 1$. Often words are stemmed, that is, suffixes and, sometimes, also prefixes are removed so that, for example, an application of Porter stemming algorithm to the words "study", "studying", and "studied" produces the term (not even an English word!) "studi." Porter stemming reduces the dictionary by about 30% and saves the memory space without seriously compromising clustering quality of English texts (handling Semitic languages like Arabic or Hebrew is much more different in this respect). In an attempt to further reduce the dimension of the vector space model often only a subset of most "meaningful" words is selected for the document–vectors construction.

A typical vector resulting from mapping a text into a finite-dimensional Euclidean space is high dimensional, sparse, and L_2 normalized. So, for example, the vector $\mathbf{a}_1$ after normalization becomes

$$\mathbf{a}_1 = \left(0, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0, 0, 0, 0, 0, 0, \frac{1}{2}, 0, 0, 0, 0, 0\right)^T.$$

This book is concerned with clustering a finite set $\mathcal{A}$ of vectors in $\mathbf{R}^n$. Motivated by IR applications it is often (but not always) assumed that the vectors are L_2 normalized and their coordinates are nonnegative real numbers.

1.2. Clustering and this book

Unlike many books on the subject this book does not attempt to cover a wide range of clustering techniques, but focuses on only three basic crisp or exclusive clustering algorithms[4] and their extensions. These algorithms are:

1. k-means,
2. Principal Direction Divisive Partitioning (PDDP),
3. Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH).

Most of the book is devoted to the k-means family of algorithms and their interconnections. Applications of the k-means algorithms equipped with a variety of distance-like functions are discussed in detail. In order to apply k-means one needs an initial partition of the data set. A common technique to address this problem is to perform multiple runs of k-means, each with a randomly selected initial centroids (which is by itself not a trivial task in a high-dimensional space), and then to select the "best" partition. We, however, advocate applications of algorithms like PDDP and BIRCH for building initial partitions for k-means.

Both PDDP and BIRCH are designed to generate partitions of large data sets. While PDDP is especially efficient with sparse and high-dimensional data (that, in particular, arises in IR applications), BIRCH is capable of handling general data sets residing in a finite-dimensional Euclidean space.

[4] That is, each object is assigned to a single cluster

The problem of the "right" choice of k, the number of clusters, is not discussed in the book.

Project 1.2.1. *For a given ASCII document D:*

1. *Build an alphabetical list of words that occur in D.*
2. *For each word compute and record the corresponding frequency.*

While the book is concerned with some mathematical techniques for clustering it is worthwhile to keep in mind that clustering is a real life problem that is difficult to cast in a mathematical formalism. El-Yaniv and Souroujon [52] illustrate the difficult choice facing clustering algorithms: "... consider a hypothetical data set containing articles by each of two authors such that half of the articles authored by each author discusses one topic, and the other half discusses another topic. There are two possible dichotomies of the data which could yield two different bi-partitions: one according to topic, and another according to writing style."

Often good clustering results depend on the "right" choice of similarity measure. Well, "a picture is worth ten thousand words," so we conclude this section with a picture of two very different object (see Figure 1.1). Should they belong in the same cluster? How to find the "right" similarity measure? What is the "goal" of a clustering procedure? While the book does not address these important questions it presents a number of techniques to generate different similarity measures and algorithms for clustering large high-dimensional data sets.

1.3. Bibliographic notes

The Cluster Hypothesis is introduced in [135]. The role of Bregman divergence in machine learning is discussed, for example, in [28, 76, 91, 92, 120]. A class of unsupervised statistical learning algorithms formulated in terms of minimizing Bregman divergences

Figure 1.1: Find the difference.

is presented in [136]. Clustering with Bregman divergence is reported in the award-winning paper [8], the extended version of the results is reported in [9]. Clustering with Csiszar divergence was introduced recently in [83–85].

For a description of document processing consult [16], for a detailed discussion see for example [58, 88, 89].

Simultaneous co-clustering of rows and columns of a matrix with nonnegative entries is considered in [122] via the information bottleneck method. Iterative double clustering built on ideas of [121] is proposed in [52]. An application of bipartite spectral graph partitioning for simultaneous co-clustering is suggested in [33]. In [15] the incremental k-means-type algorithm with Kullback–Leibler distance-like function is used to cluster rows and columns of the matrix, the batch counterpart of the algorithm is derived in [41].

A review of term weighting formulas for the vector space model is provided in [27]. For Porter stemming algorithm see [113]. For a partial

list of general clustering-related references the reader is advised to consult, for example [2–4,10,14,46,48,50,54,56,59,63–65,68,71,72,75, 79,86,87,105,106,108,124,127,137,138] .

Finally, we would like to draw the reader's attention to the new emerging field of multiway clustering. The multiway clustering is motivated mainly by computer vision applications and the existing publications are a few (see e.g. [1,62,142,149]).

2 Quadratic *k*-means algorithm

This chapter focuses on the basic version of the *k*-means clustering algorithm equipped with the quadratic Euclidean distance-like function. First, the classical batch *k*-means clustering algorithm with a general distance-like function is described and a "best representative" of a cluster, or centroid, is introduced. This completes description of the batch *k*-means with general distance-like function, and the rest of the chapter deals with *k*-means clustering algorithms equipped with the squared Euclidean distance.

Elementary properties of quadratic functions are reviewed, and the classical quadratic *k*-means clustering algorithm is stated. The following discussion of the algorithm's advantages and deficiencies results in the incremental version of the algorithm (the quadratic incremental *k*-means algorithm). In an attempt to address some of the deficiencies of batch and incremental *k*-means we merge both versions of the algorithm, and the combined algorithm is called the *k*-means clustering algorithm throughout the book.

The analysis of the computational cost associated with the merger of the two versions of the algorithm is provided and convexity properties of partitions generated by the batch *k*-means and *k*-means algorithms are discussed. Definition of "centroids" as affine subspaces of $\mathbf{R}^n$ and a brief discussion of connections between quadratic and spherical *k*-means (formally introduced in Chapter 4) complete the chapter.

2.1. Classical batch k-means algorithm

For a set of vectors $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\} \subset \mathbf{R}^n$, a prescribed subset $\mathcal{C}$ of $\mathbf{R}^n$ and a "distance" function $d(\mathbf{x}, \mathbf{a})$ define a centroid $\mathbf{c} = \mathbf{c}(\mathcal{A})$ of the set $\mathcal{A}$ as a solution of the minimization problem

$$\mathbf{c} = \arg\min \left\{ \sum_{\mathbf{a} \in \mathcal{A}} d(\mathbf{x}, \mathbf{a}),\ \mathbf{x} \in \mathcal{C} \right\}. \tag{2.1.1}$$

The quality of the set $\mathcal{A}$ is denoted by $Q(\mathcal{A})$ and is defined by

$$Q(\mathcal{A}) = \sum_{i=1}^{m} d(\mathbf{c}, \mathbf{a}), \quad \text{where } \mathbf{c} = \mathbf{c}(\mathcal{A}) \tag{2.1.2}$$

(we set $Q(\emptyset) = 0$ for convenience). Let $\Pi = \{\pi_1, \ldots, \pi_k\}$ be a partition of $\mathcal{A}$, that is,

$$\bigcup_i \pi_i = \mathcal{A}, \quad \text{and} \quad \pi_i \cap \pi_j = \emptyset \text{ if } i \neq j.$$

We abuse notations and define the quality of the partition Π by

$$Q(\Pi) = Q(\pi_1) + \cdots + Q(\pi_k). \tag{2.1.3}$$

We aim to find a partition $\Pi^{\min} = \{\pi_1^{\min}, \ldots, \pi_k^{\min}\}$ that *minimizes* the value of the objective function Q. The problem is known to be NP-hard, and we are looking for algorithms that generate "reasonable" solutions.

It is easy to see that centroids and partitions are associated as follows:

1. Given a partition $\Pi = \{\pi_1, \ldots, \pi_k\}$ of the set $\mathcal{A}$ one can define the corresponding centroids $\{\mathbf{c}(\pi_1), \ldots, \mathbf{c}(\pi_k)\}$ by:
$$\mathbf{c}(\pi_i) = \arg\min \left\{ \sum_{\mathbf{a} \in \pi_i} d(\mathbf{x}, \mathbf{a}),\ \mathbf{x} \in \mathcal{C} \right\}. \tag{2.1.4}$$
2. For a set of k "centroids" $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ one can define a partition $\Pi = \{\pi_1, \ldots, \pi_k\}$ of the set $\mathcal{A}$ by:
$$\pi_i = \left\{ \mathbf{a} \,:\, \mathbf{a} \in \mathcal{A},\ \ d(\mathbf{c}_i, \mathbf{a}) \leq d(\mathbf{c}_l, \mathbf{a}) \text{ for each } l = 1, \ldots, k \right\} \tag{2.1.5}$$