

Chapter 2

Related Work and Concepts

Abstract This chapter presents the main background knowledge relevant to the book. Sections 2.1 and 2.2 describe the areas of processing complex data and knowledge discovery in traditional databases. The task of clustering complex data is discussed in Sect. 2.3, while the task of labeling such kind of data is described in Sect. 2.4. Section 2.5 introduces the MapReduce framework, a promising tool for large scale data analysis, which has been proven to offer one valuable support to the execution of data mining algorithms in a parallel processing environment. Section 2.6 concludes the chapter.

Keywords Complex data · Moderate-to-high dimensionality data · Big data · Subspace clustering · Projected clustering · Correlation clustering · Labeling · MapReduce · Hadoop

2.1 Processing Complex Data

Database systems work efficiently with traditional numeric or textual data, but they usually do not provide complete support for complex data, such as images, videos, audio, graphs, long texts, fingerprints, geo-referenced data, among others. However, efficient methods for storing and retrieving complex data are increasingly needed [45]. Therefore, many researchers have been working to make database systems more suited to complex data processing and analysis.

The most common strategy is the manipulation of complex data based on features extracted automatically or semi-automatically from the data. This involves the application of techniques aimed at obtaining a set of features (the feature vector) to describe the complex element. Each feature is typically one value or one array of numerical values. The vector resulting from this process should properly describe the complex data, because the mining algorithms usually rely only on the features extracted to perform their tasks. It is common to find vectors containing hundreds or even thousands of features.

For example, the extraction of features from images is usually based on the analysis of colors, textures, objects' shapes and their relationship. Due to its simplicity and low computational cost, the most used color descriptor is the histogram, which counts the numbers of pixels of each color in an image [43]. The color coherence vector [57], the color correlogram [33], the metric histogram [70] and the cells histogram [66] are other well-known color descriptors. Texture corresponds to the statistical distribution of how the color varies in the neighborhood of each pixel of the image. Texture analysis is not a trivial task and it usually leads to higher computational costs than the color analysis does. Statistical methods [24, 60] analyze properties, such as granularity, contrast and periodicity to differentiate textures, while syntactic methods [24] perform this task by identifying elements in the image and analyzing their spatial arrangements. Co-occurrence matrices [32], Gabor [20, 60] and wavelet transforms [14] are examples of such methods. The descriptors of shapes commonly have the highest computational costs compared to the other descriptors, thus, they are used mainly in specific applications [58]. There are two central techniques to detect shapes: the geometric methods of edge detection [11, 60] that analyze length, curvature and signature of the edges, and the scalar methods for region detection [63] that analyze the area, "eccentricity", and "rectangularity".

Besides the feature extraction process, there are still two main problems to be addressed in order to allow an efficient management of complex data. The first is the fact that the extractors usually generate many features (hundreds or even thousands). As described in the upcoming Sect. 2.2, it impairs the existing strategies for data storage and retrieval due to the "curse of dimensionality" [10, 37, 39, 49, 56]. Therefore, dimensionality reduction techniques are vital to the success of strategies for indexing, retrieving and analyzing complex data. The second problem stems from the need to compare complex data by similarity, because it usually does not make sense to compare them by equality, as it is commonly done for traditional data [74]. Moreover, the total ordering property does not hold among complex data elements—one can only say that two elements are equal or different, since there is no explicit rule to sort the elements. This fact distinguishes complex data elements even more from the traditional elements. The access methods based on the total ordering property do not support queries involving comparisons by similarity. Therefore, a new class of access methods was created, known as the Metric Access Methods (MAM), aimed at allowing searches by similarity. Examples of such methods are the *M-tree* [17], the *Slim-tree* [71], the *DF-tree* [72] and the *DBM-tree* [74], which are considered to be dynamic, as they allow data updates without the need to rebuild the structure.

The main principle for these techniques is the representation of the data in a metric space. The similarity between two elements is calculated by a distance function acting as a metric applied to the pair of elements in the same domain. The definition of one metric space is as follows.

Definition 2.1 One **metric space** is defined as a pair $\langle \mathbb{S}, m \rangle$, where $\mathbb{S}$ is the data domain and $m : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ a distance function acting as a metric. Given any $s_1, s_2, s_3 \in \mathbb{S}$, this function must respect the following properties: (1) symmetry, $m(s_1, s_2) = m(s_2, s_1)$; (2) non-negativity, $0 < m(s_1, s_2) < \infty, \forall s_1 \neq s_2$;

(3) identity, $m(s_1, s_1) = 0$; and (4) triangle inequality, $m(s_1, s_2) \leq m(s_1, s_3) + m(s_3, s_2)$, $\forall s_1, s_2, s_3 \in \mathbb{S}$.

One particular type of metric space is the d -dimensional space, for which a distance function is defined, denoted as $\langle {}^d\mathbb{S}, m \rangle$. This case is especially interesting to this book, since it allows posing similarity queries over complex objects represented by feature vectors of equal cardinality in a multi-dimensional dataset.

2.2 Knowledge Discovery in Traditional Data

The task of *Knowledge Discovery in Databases*—*KDD* is defined as follows.

The nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [28].

This process is commonly partitioned into three steps: *Preprocessing*, *Mining* and *Result Evaluation* [61]. To obtain high-level information from raw data, the data to be analyzed is usually represented as a set of points in a d -dimensional space for which a distance function acting as a metric is specified, as described in Sect. 2.1. The attributes of the data indicate dimensions and the data objects represent points in the space, while the similarity between pairs of objects is measured in terms of the respective distance function applied to the data space.

With the increasing quantity and complexity of the data generated or collected in digital systems, the task of *Preprocessing* has become essential to the whole KDD process [65]. In this step, the data are reduced and prepared by cleaning, integrating, selecting and transforming the objects to the subsequent mining step. A major problem to be minimized is the “curse of dimensionality” [10, 37, 39, 49, 56], a term referring to the fact that increasing the number of attributes in the objects quickly leads to significant degradation of the performance and also of the accuracy of existing techniques to access, store and process data. This occurs because data represented in high dimensional spaces tend to be extremely sparse and all the distances between any pair of points tend to be very similar, with respect to various distance functions and data distributions [10, 39, 56]. Dimensionality reduction is the most common technique applied to minimize this problem. It aims at obtaining a set of relevant and non-correlated attributes that allow representing the data in a space of lower dimensionality with minimum loss of information. The existing approaches are: feature selection, which discards among the original attributes the ones that contribute with less information to the data objects; and feature extraction, which creates a reduced set of new features, formed by linear combinations of the original attributes, able to represent the data with little loss of information [19].

The mining task is a major step in KDD. It involves the application of data mining algorithms chosen according to the user’s objectives. Such tasks are classified as: predictive tasks, which seek a model to predict the value of an attribute based on the values of other attributes, by generalizing known examples, and descriptive tasks, which look for patterns that describe the intrinsic data behavior [64].

Classification is a major predictive mining task. It considers the existence of a training set with records classified according to the value of an attribute (target attribute or class) and a test set, in which the class of each record is unknown. The main goal is to predict the values of the target attribute (class) in the database records to be classified. The algorithms perform data classification by defining rules to describe correlations between the class attribute and the others. Examples of such algorithms are genetic algorithms [9, 82] and algorithms based on decision trees [13], on neural networks [25] or on the Bayes theorem (Bayesian classification) [81].

Clustering is an important descriptive task. Han and Kamber define it as follows.

The process of grouping the data into classes or clusters, so that objects within a cluster have high similarity in comparison to one another but are very dissimilar to objects in other clusters [31].

Traditional clustering algorithms are commonly divided into: (1) hierarchical algorithms, which define a hierarchy between the clusters in a process that may be initiated with a single cluster, recursively partitioned in follow-up steps (*top-down*), or considering at first that each data object belongs to a distinct cluster, recursively merging the clusters latter (*bottom-up*); or (2) partitioning algorithms, which divide η objects into k clusters, $k \leq \eta$, such that each object belongs to at most one cluster and each cluster contains at least one object. Examples of well-known clustering methods are k -Means [42, 44, 67], k -Harmonic Means [80], DBSCAN [26] and STING [75].

The last step in the process of knowledge discovery is the result evaluation. At this stage, the patterns discovered in the previous step are interpreted and evaluated. If the patterns refer to satisfactory results (valid, novel, potentially useful and ultimately understandable), the knowledge is consolidated. Otherwise, the process returns to one previous stage to improve the results.

2.3 Clustering Complex Data

Complex data are usually represented by vectors with hundreds or even thousands of features in a multi-dimensional space, as described in Sect. 2.1. Each feature represents one dimension of the space. Due to the curse of dimensionality, traditional clustering methods, like k -Means [42, 44, 67], k -Harmonic Means [80], DBSCAN [26] and STING [75], are commonly inefficient and ineffective for these data [4–7]. The main factor that leads to their efficiency problem is that they often have super-linear computational complexity on both cardinality and dimensionality. Traditional methods are also often ineffective when applied to high dimensional data, since the data tend to be very sparse in the multi-dimensional space and the distances between any pair of points usually become very similar to each other, regarding several data distributions and distance functions [10, 39, 49, 56]. Thus, traditional methods do not solve the problem of clustering large, complex datasets [4–7, 23, 39, 49, 56, 73].

Dimensionality reduction methods minimize the effects of the dimensionality curse by finding a new set of orthogonal dimensions, of cardinality smaller than the original set's one, composed of non-correlated dimensions relevant to characterize the data. The elements of this set can be original dimensions, in the case of feature selection methods, or linear combinations of them, for feature extraction methods. Notice however that only global correlations are identified by such methods. In other words, dimensionality reduction methods look for correlations that occur for *all* dataset elements regarding a set of dimensions. Nevertheless, high dimensional data often present correlations local to subsets of the data elements and dimensions [23, 73]. Thus, distinct groups of data points correlated with different sets of dimensions may exist. Many of these correlations can also be non-linear. Therefore, it is clear that traditional dimensionality reduction techniques do not identify all possible correlations, as they evaluate correlations regarding the entire dataset, and thus, when used as a preprocessing step for traditional clustering, they do not solve the problem of clustering large, complex datasets [4–7, 23].

Since high dimensional data often present correlations local to subsets of elements and dimensions, the data are likely to present clusters that only exist in subspaces of the original data space. In other words, although high dimensional data usually do not present clusters in the space formed by all dimensions, the data tend to form clusters when the points are projected into subspaces generated by reduced sets of original dimensions or linear combinations of them. Moreover, different clusters may be formed in distinct subspaces. Several recent studies support this idea and a well-known survey on this area is presented at [39].

Figure 2.1 exemplifies the existence of clusters in subspaces of four 3-dimensional databases over the axes $E = \{x, y, z\}$. Figure 2.1a shows a 3-dimensional dataset projected onto axes x and y , while Fig. 2.1b shows the same dataset projected onto axes x and z . There exist two clusters in this data, C_1 and C_2 . None of the clusters present a high density of points in the 3-dimensional space, but each cluster is a dense, elliptical object in one subspace. Thus, the clusters exist in subspaces only. Cluster C_1 exists in the subspace formed by axes x and z , while cluster C_2 exists in the subspace $\{x, y\}$. Besides elliptical clusters, real data may have clusters that assume *any* shape in their respective subspaces. The clusters must only be dense in that subspace. To illustrate this fact, we present ‘star-shaped’ and ‘triangle-shaped’ clusters in another dataset (Fig. 2.1c: x - y projection; Fig. 2.1d: x - z projection) and ‘curved’ clusters in one third example dataset (Fig. 2.1e: x - y projection; Fig. 2.1f: x - z projection).

Such clusters may also exist in subspaces formed by linear combinations of original axes. That is, clusters like the ones in our previous examples may be arbitrarily *rotated* in the space, thus not being aligned to the original axes. For example, Fig. 2.1g and h respectively plot x - y and x - z projections of one last 3-dimensional example dataset. Similarly to clusters C_1 and C_2 , the clusters in this data are also dense, elliptical objects in subspaces, but, in this case, the subspaces are planes generated by linear combinations of the axes $\{x, y, z\}$. Traditional clustering is likely to fail when analyzing this data, as each cluster is spread over an axis (generated by linear combinations of the original axes) in the 3-dimensional space. Also, dimensionality

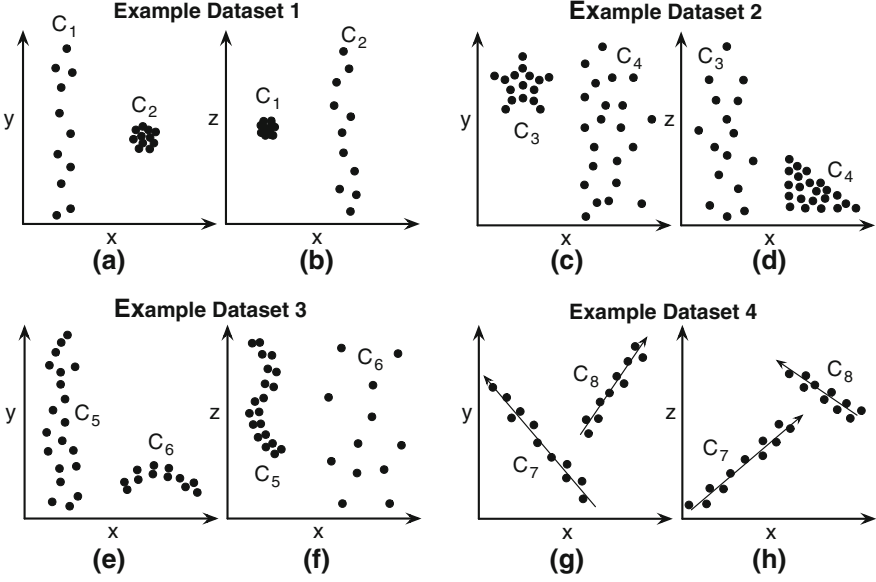


Fig. 2.1 x - y and x - z projections of four 3-dimensional datasets over axes $\{x, y, z\}$. From **a** to **f**: clusters in the subspaces $\{x, y\}$ and $\{x, z\}$. **g** and **h**: clusters in subspaces formed by linear combinations of $\{x, y, z\}$

reduction applied to the entire dataset does not help, as no 1- or 2-dimensional projection, axis aligned or not, keeps the clusters apart. This problem tends to be even worse in spaces of higher dimensionality. In fact, there is an interdependence between traditional clustering and dimensionality reduction that prevents them from solving the problem of clustering complex data. It is a fact that traditional clustering depends on a prior dimensionality reduction to analyze this kind of data. On the other hand, dimensionality reduction treats only global correlations, the ones that happen with regard to all data elements. Correlations local to subsets of the data cannot be identified without knowing the respective subsets. In other words, clustering complex data is not possible due to the high dimensionality, but the dimensionality reduction is incomplete without the prior knowledge of the data clusters where local correlations occur.

The most common strategy used to untangle this knotty problem is to unify both tasks, clustering and dimensionality reduction, creating a single task. Several methods have used this idea in order to look for clusters together with the subspaces where the clusters exist. According to a well-known survey [39], such methods differ from each other in two major aspects: (1) the search strategy used, which can be *top-down* or *bottom-up*, and (2) the characteristics of the clusters sought.

The *bottom-up* algorithms usually rely on a property named *downward closure* or *monotonicity*. This property, valid for some criteria of clustering characterization, warrants that: if there is at least one cluster in a d -dimensional dataset, at least one

cluster will stand out when the data are projected into each of the possible subspaces formed by the original dimensions [7, 38, 39, 56]. As an example, this property is valid when the criterion used for clustering characterization is the *minimum density threshold*. By this criterion, one part of the data space contains a cluster if its density of points meet or exceed the minimum bound specified. The density of points never reduces when data are projected onto subspaces of the original space. Thus, if there is one part of a d -dimensional space whose density of points is sufficient to characterize a cluster, it is possible to affirm that at least one cluster will stand out (i.e., at least one space region will be dense enough) when the data are projected into any subspace formed by the original dimensions.

Based on a criterion of clustering characterization in which the *downward closure* property applies, *bottom-up* algorithms assume that: if a cluster exists in a space of high dimensionality, it has to exist or to be part of some cluster in all subspaces of lower dimensionality formed by original dimensions [7, 39, 56]. With that in mind, these methods start analyzing low dimensional subspaces, usually 1-dimensional ones, in order to identify the subspaces that contain clusters. The subspaces selected are then united in a recursive procedure that allows the identification of subspaces of higher dimensionality in which clusters also exist. Various techniques are used to spot and to prune subspaces without clusters so that they can be ignored in the process, minimizing the computational cost involved, but, in general, the main shortcomings of *bottom-up* methods are: (1) they often have super-linear computational complexity or even exponential complexity regarding the dimensionality of the subspaces analyzed, and (2) fixed density thresholds are commonly used, assuming that clusters in high dimensional spaces are as dense as clusters in subspaces of smaller dimensionality, an assumption that is unlikely to be true in several cases.

The *top-down* algorithms assume that the analysis of the space with all dimensions can identify patterns that lead to the discovery of clusters existing in lower dimensional subspaces only [56]. This assumption is known in literature as the *locality assumption* [39]. After identifying a pattern, the distribution of points surrounding the pattern in the space with all dimensions is analyzed to define whether or not the pattern refers to a cluster and the subspace in which the possible cluster is better characterized. The main drawbacks of *top-down* algorithms are: (1) they often have super-linear computational complexity, though not exponential complexity, with regard to the data dimensionality, and; (2) there is no guarantee that the analysis of the data distribution in the space with all dimensions is always sufficient to identify clusters that exist in subspaces only.

Clustering algorithms for high dimensional data also differ from each other in the characteristics of the clusters that they look for. Some algorithms look for clusters that form a dataset partition, together with a set of outliers, while other algorithms partition the database regarding specific subspaces, so that the same data element can be part of two or more clusters that overlap with each other in the space with all dimensions, as long as the clusters are formed in different subspaces [56]. Finally, the subspaces analyzed by these methods may be limited to subsets of the original dimensions or may not be limited to them, thus including subspaces formed by linear combinations of the original dimensions [39].

Subspace clustering algorithms aim at analyzing projections of the dataset into spaces formed by subsets of the *original dimensions only*. Given a subspace and its corresponding data projection, these algorithms operate similarly to traditional clustering algorithms, partitioning the data into disjoint sets of elements, named **subspace clusters**, and one set of outliers. A data point can belong to more than one subspace cluster, as long as the clusters exist in different subspaces. Therefore, subspace clusters are *not necessarily disjoint*. Examples of subspace clustering algorithms are: CLIQUE [5, 7], ENCLUS [16], SUBCLU [34], FIRES [38], P3C [47, 48] and STATPC [46].

Projected clustering algorithms aim at partitioning the dataset into *disjoint* sets of elements, named **projected clusters**, and one set of outliers. A subspace formed by the *original dimensions* is assigned to each cluster, and the cluster elements are densely grouped, when projected into the respective subspace. Examples of projected clustering algorithms in literature are: PROCLUS [6], DOC/FASTDOC [59], PreDeCon [12], COSA [29], FINDIT [77], HARP [78], EPCH [51, 52], SSPC [79], P3C [47, 48] and LAC [8, 22, 23].

Correlation clustering algorithms aim at partitioning the database in a manner analogous to what occurs with projected clustering techniques—to identify *disjoint* clusters and one set of outliers. However, the clusters identified by such methods, named **correlation clusters**, are composed of densely grouped elements in subspaces formed by the original dimensions of the database, *or by their linear combinations*. Examples of correlation clustering algorithms found in literature are: ORCLUS [3, 4], 4C [12], CURLER [73], COPAC [1] and CASH [2]. Notice that, to the best of our knowledge, CURLER is the only method found in literature that spots clusters formed by non-linear, local correlations, besides the ones formed by linear, local-correlations.

2.4 Labeling Complex Data

In this section, we assume that the clustering algorithms aimed at analyzing complex data, such as the ones cited in the previous section, can also serve as a basis to perform one *distinct* data mining task—the task of labeling large sets of complex objects, as we discuss in the upcoming Chap. 6. For that reason, the following paragraphs introduce some background knowledge related to the task of labeling complex data, i.e., the task of analyzing a given collection of complex objects, in which a few objects are labeled with keywords, in order to spot appropriate labels for the remaining majority of unlabeled objects.

Specifically, the task of labeling is one predictive data mining task that considers the existence of a training set containing records labeled with keywords and a test set, in which the labels of each record are unknown. Its main goal is to assign appropriate keywords to the database records to be labeled. Unlike other data mining tasks, the task of labeling is not completely defined in the literature and slight conceptual divergences exist in the definitions provided by distinct authors, while some authors

consider that labeling is one type of classification or use other names to refer to it (e.g., captioning). In this book, we consider that labeling refers to one generalized version of the classification task, in which the restriction of mandatorily assigning one and only one label to every data object does not exist. Specifically, we assume that labeling generalizes the task of classification with regard to at most three aspects: (1) it may consider that the dataset commonly contains objects that differ too much from the labeled training examples, which should be returned to the user as outliers that potentially deserve a new label of their own; (2) it may allow any object to receive more than one appropriate label; and (3) it may use hierarchies of labels, in a way that each object can be assigned to entire paths in the hierarchy, instead of being linked to individual labels only.

In the scope of complex data, labeling has been mostly applied to image datasets and also to sets of image regions segmented or arbitrarily extracted from larger images. There is an extensive body of work on the labeling of unlabeled regions extracted from partially labeled images in the computer vision field, such as image segmentation and region classification [41, 62, 69]. The Conditional Random Fields (CRF) and boosting approach [62] shows a competitive accuracy for multi-label labeling and segmentation, but it is relatively slow and requires many training examples. The KNN classifier [69] may be the fastest way for image region labeling, but it is not robust against outliers. Also, the Empirical Bayes approach [41] proposes to learn contextual information from unlabeled data. However, it may be difficult to learn the context from several types of complex data, such as from satellite images.

The principle Random Walks with Restarts (RWR) [68] has served as a basis to other labeling methods. The idea is to perform labeling by creating a graph to represent the input complex objects to be labeled, the given example labels and the similarities existing between the objects. Then, random walks in this graph allow spotting the most appropriate labels for the remaining unlabeled objects. In general, RWR consists into performing walks in a graph according to the following strategy: a random walker starts a walk from a vertex V of the graph, and, at each time step, the walker either goes back to the initial vertex V , with a user-defined probability c , or it goes to a randomly chosen vertex that shares an edge with the current vertex, with probability $1 - c$. The intuition is that this procedure provides an appropriate relevance score between any pair of graph nodes, since the steady state probability that a random walker will find itself in a vertex V' , always restarting the walk from a vertex V , is a way to measure the closeness between V and V' . For labeling, RWR is commonly used to measure the closeness between the graph nodes that represent data objects and the ones that represent keywords.

GCap [54] is one of the most famous labeling methods based on random walks with restarts. It proposes a graph-based strategy for automatic image labeling that can also be applied to any set of multimedia objects. GCap represents images and label keywords by multiple layers of nodes in a graph and captures the similarities between pairs of images by creating edges to link the nodes that refer to similar images. The known labels become links between the respective images and keywords. This procedure creates a tri-partite graph that represents the input images and labels, besides the existing similarities between the images. Given an image

node of interest, random walks with restarts are used to perform proximity queries in this graph, allowing GCap to automatically find the best annotation keyword for the respective image. Unfortunately, GCap remains rather inefficient, since it searches for the nearest neighbors of every image in the feature space to create edges between similar image nodes, and this operation is super-linear even with the speed up offered by approximate nearest-neighbor finding algorithms (e.g., the ANN Library [50]).

2.5 MapReduce

The large amounts of data collected by enterprises are accumulating data, and today it is already feasible to have *Terabyte*- or even *Petabyte-scale* datasets that must be submitted to data mining processes (e.g., Twitter crawl: > 12 TB, Yahoo! operational data: 5 *Petabytes* [27]), such as the processes of clustering and labeling complex data. However, the use of serial data mining algorithms, like the ones described in the previous sections, to analyze such huge amounts of data is clearly an impractical task. Just to read a single Terabyte of data (at 5 GB/min on a single modern eSATA disk) one takes more than 3 hours. Therefore, to improve the existing serial data mining methods in order to make them run efficiently in parallel is nowadays extremely desirable. With that in mind, this section describes the MapReduce framework, a promising tool for large-scale, parallel data analysis, which has been proving to offer one valuable support to the execution of data mining algorithms in a parallel processing environment.

MapReduce is a programming framework [21] fostered by Google¹ to process large-scale data in a massively parallel way. MapReduce has two major advantages: the programmer is oblivious of the details related to the data storage, distribution, replication, load balancing, etc.; and furthermore, it adopts the familiar concept of functional programming. The programmer needs to specify only two functions, a *map* and a *reduce*. The typical framework is as follows [40]: (a) the *map* stage passes over the input file and outputs (key, value) pairs; (b) the *shuffling* stage transfers the mappers output to the reducers based on the key, and; (c) the *reduce* stage processes the received pairs and outputs the final result. Due to its scalability, simplicity and the low cost to build large clouds of computers, MapReduce is a very promising tool for large scale, parallel data analysis, a fact that is already being reflected in the academia (e.g., [18, 35, 36, 55]).

Hadoop is the open source implementation of MapReduce. Hadoop provides the Hadoop Distributed File System (HDFS) [30], HBase [76], which is a way to efficiently store and handle semi-structured data as Google's BigTable storage system [15], and PIG, a high level language for data analysis [53].

¹ www.google.com

2.6 Conclusions

In this chapter we presented an overview of the basic concepts used in the book. We described the research areas of processing complex data and knowledge discovery in traditional databases, besides the main factors that distinguish the tasks of traditional clustering and clustering complex data. The task of labeling large sets of complex objects was also discussed. Finally, we introduced the MapReduce framework, a promising tool for large scale data analysis, which has been proving to offer one valuable support to the execution of data mining algorithms in a parallel processing environment. The next chapter describes some of the most relevant clustering methods available in literature for multi-dimensional data with more than five or so dimensions, which look for clusters formed by local correlations.

References

1. Achtert, E., Böhm, C., Kriegel, H.P., Kröger, P., Zimek, A.: Robust, complete, and efficient correlation clustering. SDM, USA, In (2007)
2. Achtert, E., Böhm, C., David, J., Kröger, P., Zimek, A.: Global correlation clustering based on the hough transform. Stat. Anal. Data Min. **1**, 111–127 (2008). doi:[10.1002/sam.v1:3](https://doi.org/10.1002/sam.v1:3)
3. Aggarwal, C., Yu, P.: Redefining clustering for high-dimensional applications. IEEE TKDE **14**(2), 210–225 (2002). doi:[10.1109/69.991713](https://doi.org/10.1109/69.991713)
4. Aggarwal, C.C., Yu, P.S.: Finding generalized projected clusters in high dimensional spaces. SIGMOD Rec. **29**(2), 70–81 (2000). doi:[10.1145/335191.335383](https://doi.org/10.1145/335191.335383)
5. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. SIGMOD Rec. **27**(2), 94–105 (1998). doi:[10.1145/276305.276314](https://doi.org/10.1145/276305.276314)
6. Aggarwal, C.C., Wolf, J.L., Yu, P.S., Procopiuc, C., Park, J.S.: Fast algorithms for projected clustering. SIGMOD Rec. **28**(2), 61–72 (1999). doi:[10.1145/304181.304188](https://doi.org/10.1145/304181.304188)
7. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data. Data Min. Knowl. Discov. **11**(1), 5–33 (2005). doi:[10.1007/s10618-005-1396-1](https://doi.org/10.1007/s10618-005-1396-1)
8. Al-Razgan, M., Domeniconi, C.: Weighted clustering ensembles. In: Ghosh, J., Lambert, D., Skillicorn, D.B., Srivastava, J. (eds.) SDM. SIAM (2006).
9. Ando, S., Iba, H.: Classification of gene expression profile using combinatory method of evolutionary computation and machine learning. Genet. Program Evolvable Mach. **5**, 145–156 (2004). doi:[10.1023/B:GENP.0000023685.83861.69](https://doi.org/10.1023/B:GENP.0000023685.83861.69)
10. Beyer, K.S., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is “nearest neighbor” meaningful? In: ICDT, pp. 217–235. UK (1999).
11. Blicher, A.P.: Edge detection and geometric methods in computer vision (differential topology, perception, artificial intelligence, low-level). Ph.D. thesis, University of California, Berkeley (1984). AAI8512758.
12. Bohm, C., Kailing, K., Kriegel, H.P., Kroger, P.: Density connected clustering with local subspace preferences. In: ICDM '04: Proceedings of the 4th IEEE International Conference on Data Mining, pp. 27–34. IEEE Computer Society, Washington, DC, USA (2004).
13. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth, Belmont (1984)
14. Chan, T.F., Shen, J.: Image processing and analysis-variational, PDE, wavelet, and stochastic methods. SIAM (2005).

15. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: a distributed storage system for structured data. In: USENIX'06. Berkeley, CA, USA (2006).
16. Cheng, C.H., Fu, A.W., Zhang, Y.: Entropy-based subspace clustering for mining numerical data. In: KDD, pp. 84–93. NY, USA (1999). doi:<http://doi.acm.org/10.1145/312129.312199>
17. Ciaccia, P., Patella, M., Zezula, P.: M-tree: an efficient access method for similarity search in metric spaces. In: The VLDB Journal, pp. 426–435 (1997).
18. Cordeiro, R.L.F., Traina Jr., C., Traina, A.J.M., López, J., Kang, U., Faloutsos, C.: Clustering very large multi-dimensional datasets with mapreduce. In: Apté, C., Ghosh, J., Smyth, P. (eds.) KDD, pp. 690–698. ACM (2011).
19. Dash, M., Liu, H., Yao, J.: Dimensionality reduction for unsupervised data. In: Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97), pp. 532–539 (1997).
20. Daugman, J.G.: Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. J. Opt. Soc. Am. A **2**, 1160–1169 (1985). doi:[10.1364/JOSAA.2.001160](https://doi.org/10.1364/JOSAA.2.001160)
21. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. OSDI (2004).
22. Domeniconi, C., Papadopoulos, D., Gunopulos, D., Ma, S.: Subspace clustering of high dimensional data. In: Berry, M.W., Dayal, U., Kamath, C., Skillicorn, D.B. (eds.) SDM (2004).
23. Domeniconi, C., Gunopulos, D., Ma, S., Yan, B., Al-Razgan, M., Papadopoulos, D.: Locally adaptive metrics for clustering high dimensional data. Data Min. Knowl. Discov. **14**(1), 63–97 (2007). doi:[10.1007/s10618-006-0060-8](https://doi.org/10.1007/s10618-006-0060-8)
24. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. Wiley, New York (2001)
25. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley-Interscience, New York (2000)
26. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, pp. 226–231 (1996).
27. Fayyad, U.: A data miner's story-getting to know the grand challenges. In: Invited Innovation Talk, KDD (2007). Slide 61. Available at: http://videolectures.net/kdd07_fayyad_dms/
28. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery: an overview. In: Advances in Knowledge Discovery and Data Mining, pp. 1–34 (1996).
29. Friedman, J.H., Meulman, J.J.: Clustering objects on subsets of attributes (with discussion). J. Roy. Stat. Soc. B **66**(4), 815–849 (2004). doi:[ideas.repec.org/a/bla/jorssb/v66y2004i4p815-849.html](https://doi.org/ideas.repec.org/a/bla/jorssb/v66y2004i4p815-849.html)
30. Hadoop information. <http://hadoop.apache.org/>
31. Han, J., Kamber, M.: Data Mining: Concepts and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2006)
32. Haralick, R.M., Shanmugam, K., Dinstein, I.: Textural features for image classification. Syst. Man Cybern. IEEE Trans. **3**(6), 610–621 (1973). doi:[10.1109/TSMC.1973.4309314](https://doi.org/10.1109/TSMC.1973.4309314)
33. Huang, J., Kumar, S., Mitra, M., Zhu, W.J., Zabih, R.: Image indexing using color correlograms. In: Proceedings of 1997 IEEE Computer Society Conference on Computer Vision and, Pattern Recognition, pp. 762–768 (1997). doi:[10.1109/CVPR.1997.609412](https://doi.org/10.1109/CVPR.1997.609412).
34. Kailing, K., Kriegel, H.: Kroger, P. Density-connected subspace clustering for highdimensional data (2004).
35. Kang, U., Tsourakakis, C., Faloutsos, C.: Pegasus: a peta-scale graph mining system-implementation and observations. ICDM (2009).
36. Kang, U., Tsourakakis, C., Appel, A.P., Faloutsos, C., Leskovec, J.: Radius plots for mining tera-byte scale graphs: algorithms, patterns, and observations. SDM (2010).
37. Korn, F., Pagel, B.U., Faloutsos, C.: On the 'dimensionality curse' and the 'self-similarity blessing'. IEEE Trans. Knowl. Data Eng. (TKDE) **13**(1), 96–111 (2001). doi:[10.1109/69.908983](https://doi.org/10.1109/69.908983)
38. Kriegel, H.P., Kröger, P., Renz, M., Wurst, S.: A generic framework for efficient subspace clustering of high-dimensional data. In: ICDM, pp. 250–257. Washington, USA (2005). doi:[http://dx.doi.org/10.1109/ICDM.2005.5](https://doi.org/10.1109/ICDM.2005.5)

39. Kriegel, H.P., Kröger, P., Zimek, A.: Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM TKDD* **3**(1), 1–58 (2009). doi:[10.1145/1497577.1497578](https://doi.org/10.1145/1497577.1497578)
40. Lämmel, R.: Google's mapreduce programming model-revisited. *Sci. Comput. Program.* **70**, 1–30 (2008)
41. Lazebnik, S., Raginsky, M.: An empirical bayes approach to contextual region classification. In: *CVPR*, pp. 2380–2387. IEEE (2009).
42. Lloyd, S.: Least squares quantization in pcm. *Inf. Theory IEEE Trans.* **28**(2), 129–137 (1982). doi:[10.1109/TIT.1982.1056489](https://doi.org/10.1109/TIT.1982.1056489)
43. Long, F., Zhang, H., Feng, D.D.: Fundamentals of content-based image retrieval. In: *Multimedia Information Retrieval and Management*. Springer (2002).
44. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: *Cam, L.M.L., Neyman, J. (eds.) Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297. University of California Press (1967).
45. Mehrotra, S., Rui, Y., Chakrabarti, K., Ortega, M., Huang, T.S.: Multimedia analysis and retrieval system. In: *Proceedings of 3rd International Workshop on Multimedia. Information Systems*, pp. 25–27 (1997).
46. Moise, G., Sander, J.: Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In: *KDD*, pp. 533–541 (2008).
47. Moise, G., Sander, J., Ester, M.: P3C: a robust projected clustering algorithm. In: *ICDM*, pp. 414–425. IEEE Computer Society (2006).
48. Moise, G., Sander, J., Ester, M.: Robust projected clustering. *Knowl. Inf. Syst.* **14**(3), 273–298 (2008). doi:[10.1007/s10115-007-0090-6](https://doi.org/10.1007/s10115-007-0090-6)
49. Moise, G., Zimek, A., Kröger, P., Kriegel, H.P., Sander, J.: Subspace and projected clustering: experimental evaluation and analysis. *Knowl. Inf. Syst.* **21**(3), 299–326 (2009)
50. Mount, D.M., Arya, S.: Ann: a library for approximate nearest neighbor searching. <http://www.cs.umd.edu/mount/ANN/>
51. Ng, E.K.K., Fu, A.W.: Efficient algorithm for projected clustering. In: *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*, p. 273. IEEE Computer Society, Washington, DC, USA (2002).
52. Ng, E.K.K., chee Fu, A.W., Wong, R.C.W.: Projective clustering by histograms. *TKDE* **17**(3), 369–383 (2005). doi:[10.1109/TKDE.2005.47](https://doi.org/10.1109/TKDE.2005.47)
53. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig latin: a not-so-foreign language for data processing. In: *SIGMOD '08*, pp. 1099–1110 (2008).
54. Pan, J.Y., Yang, H.J., Faloutsos, C., Duygulu, P.: Gcap: graph-based automatic image captioning. In: *CVPRW '04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition, Workshop (CVPRW'04) vol. 9*, p. 146 (2004).
55. Papadimitriou, S., Sun, J.: Disco: distributed co-clustering with map-reduce. *ICDM* (2008).
56. Parsons, L., Haque, E., Liu, H.: Subspace clustering for high dimensional data: a review. *SIGKDD Explor. Newsl* **6**(1), 90–105 (2004). doi:[10.1145/1007730.1007731](https://doi.org/10.1145/1007730.1007731)
57. Pass, G., Zabih, R., Miller, J.: Comparing images using color coherence vectors. In: *ACM Multimedia*, pp. 65–73 (1996).
58. Pentland, A., Picard, R.W., Sclaroff, S.: Photobook: tools for content-based manipulation of image databases. In: *Storage and Retrieval for Image and Video Databases (SPIE)*, pp. 34–47 (1994).
59. Procopiuc, C.M., Jones, M., Agarwal, P.K., Murali, T.M.: A monte carlo algorithm for fast projective clustering. In: *SIGMOD*, pp. 418–427. USA (2002). doi:[http://doi.acm.org/10.1145/564691.564739](https://doi.org/http://doi.acm.org/10.1145/564691.564739)
60. Rangayyan, R.M.: *Biomedical Image Analysis*. CRC Press, Boca Raton (2005)
61. Rezende, S.O.: *Sistemas Inteligentes: Fundamentos e Aplicações*. Ed , Manole Ltda (2002). (in Portuguese)
62. Shotton, J., Winn, J.M., Rother, C., Criminisi, A.: TextonBoost: joint appearance, shape and context modeling for multi-class object recognition and segmentation. In: *Leonardis, A.,*

- Bischof, H., Pinz A. (eds.) ECCV (1), Lecture Notes in Computer Science, vol. 3951, pp. 1–15. Springer (2006).
63. Sonka, M., Hlavac, V., Boyle, R.: Image Processing: Analysis and Machine Vision, 2nd edn. Brooks/Cole Pub Co, Pacific Grove (1998)
 64. Sousa, E.P.M.: Identificação de correlações usando a teoria dos fractais. Ph.D. Dissertation, Computer Science Department–ICMC, University of São Paulo–USP, São Carlos, Brazil (2006). (in Portuguese).
 65. Sousa, E.P.: Caetano Traina, J., Traina, A.J., Wu, L., Faloutsos, C.: A fast and effective method to find correlations among attributes in databases. *Data Min. Knowl. Discov.* **14**(3), 367–407 (2007). doi:[10.1007/s10618-006-0056-4](https://doi.org/10.1007/s10618-006-0056-4)
 66. Stehling, R.O., Nascimento, M.A., Falcão, A.X.: Cell histograms versus color histograms for image representation and retrieval. *Knowl. Inf. Syst.* **5**, 315–336 (2003). doi:[10.1007/s10115-003-0084-y](https://doi.org/10.1007/s10115-003-0084-y). <http://portal.acm.org/citation.cfm?id=959128.959131>
 67. Steinhaus, H.: Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci.* **1**, 801–804 (1956). (in French).
 68. Tong, H., Faloutsos, C., Pan, J.Y.: Random walk with restart: fast solutions and applications. *Knowl. Inf. Syst.* **14**, 327–346 (2008). doi:[10.1007/s10115-007-0094-2](https://doi.org/10.1007/s10115-007-0094-2). <http://portal.acm.org/citation.cfm?id=1357641.1357646>
 69. Torralba, A.B., Fergus, R., Freeman, W.T.: 80 million tiny images: a large data set for non-parametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(11), 1958–1970 (2008)
 70. Traina, A.J.M., Traina, C., Bueno, J.M., Chino, F.J.T., Azevedo-Marques, P.: Efficient content-based image retrieval through metric histograms. *World Wide Web* **6**, 157–185 (2003). doi:[10.1023/A:1023670521530](https://doi.org/10.1023/A:1023670521530).
 71. Traina Jr, C., Traina, A.J.M., Seeger, B., Faloutsos, C.: Slim-trees: high performance metric trees minimizing overlap between nodes. In: Zaniolo, C., Lockemann, P.C., Scholl, M.H., Grust, T. (eds.) *International Conference on Extending Database Technology (EDBT)*. Lecture Notes in Computer Science, vol. 1777, pp. 51–65. Springer, Konstanz, Germany (2000).
 72. Traina Jr, C., Traina, A.J.M., Santos Filho, R.F., Faloutsos, C.: How to improve the pruning ability of dynamic metric access methods. In: *International Conference on Information and Knowledge Management (CIKM)*, pp. 219–226. ACM Press, McLean, VA, USA (2002).
 73. Tung, A.K.H., Xu, X., Ooi, B.C.: Curler: finding and visualizing nonlinear correlation clusters. In: *SIGMOD*, pp. 467–478 (2005). doi:[http://doi.acm.org/10.1145/1066157.1066211](https://doi.org/http://doi.acm.org/10.1145/1066157.1066211)
 74. Vieira, M.R., Traina Jr, C., Traina, A.J.M., Chino, F.J.T.: Dbm-tree: a dynamic metric access method sensitive to local density data. In: Lifschitz, S. (ed.) *Brazilian Symposium on Databases (SBBD)*, vol. 1, pp. 33–47. SBC, Brasília, DF (2004)
 75. Wang, W., Yang, J., Muntz, R.: Sting: a statistical information grid approach to spatial data mining. In: *Vldb*, pp. 186–195 (1997).
 76. Wiki: <http://wiki.apache.org/hadoop/hbase>. Hadoop’s Bigtable-like structure
 77. Woo, K.G., Lee, J.H., Kim, M.H., Lee, Y.J.: Findit: a fast and intelligent subspace clustering algorithm using dimension voting. *Inf. Softw. Technol.* **46**(4), 255–271 (2004)
 78. Yip, K.Y., Ng, M.K.: Harp: a practical projected clustering algorithm. *IEEE Trans. on Knowl. Data Eng.* **16**(11), 1387–1397 (2004). doi:[http://dx.doi.org/10.1109/TKDE.2004.74](https://doi.org/http://dx.doi.org/10.1109/TKDE.2004.74). Member-David W. Cheung
 79. Yip, K.Y., Cheung, D.W., Ng, M.K.: On discovery of extremely low-dimensional clusters using semi-supervised projected clustering. In: *ICDE*, pp. 329–340. Washington, USA (2005). doi:[http://dx.doi.org/10.1109/ICDE.2005.96](https://doi.org/http://dx.doi.org/10.1109/ICDE.2005.96)
 80. Zhang, B., Hsu, M., Dayal, U.: K-harmonic means-a spatial clustering algorithm with boosting. In: Roddick, J.F., Hornsby, K. (eds.) *TSDM. Lecture Notes in Computer Science*, vol. 2007, pp. 31–45. Springer (2000).
 81. Zhang, H.: The optimality of naive Bayes. In: V. Barr, Z. Markov (eds.) *FLAIRS Conference*. AAAI Press (2004). <http://www.cs.unb.ca/profs/hzhang/publications/FLAIRS04ZhangH.pdf>
 82. Zhou, C., Xiao, W., Tirpak, T.M., Nelson, P.C.: Evolving accurate and compact classification rules with gene expression programming. *IEEE Trans. Evol. Comput.* **7**(6), 519–531 (2003)

Data Mining in Large Sets of Complex Data

Ferreira Cordeiro, R.L.; Faloutsos, C.; Traina Júnior, C.

2013, XI, 116 p. 37 illus., 25 illus. in color., Softcover

ISBN: 978-1-4471-4889-0