Systematic Review of Clustering High-Dimensional and Large Datasets

DIVYA PANDOVE, Thapar University
SHIVANI GOEL, Bennett University
RINKLE RANI, Thapar University

Technological advancement has enabled us to store and process huge amount of data in relatively short spans of time. The nature of data is rapidly changing, particularly its dimensionality is more commonly multi- and high-dimensional. There is an immediate need to expand our focus to include analysis of high-dimensional and large datasets. Data analysis is becoming a mammoth task, due to incremental increase in data volume and complexity in terms of heterogony of data. It is due to this dynamic computing environment that the existing techniques either need to be modified or discarded to handle new data in multiple high-dimensions. Data clustering is a tool that is used in many disciplines, including data mining, so that meaningful knowledge can be extracted from seemingly unstructured data. The aim of this article is to understand the problem of clustering and various approaches addressing this problem. This article discusses the process of clustering from both microviews (data treating) and macroviews (overall clustering process). Different distance and similarity measures, which form the cornerstone of effective data clustering, are also identified. Further, an in-depth analysis of different clustering approaches focused on data mining, dealing with large-scale datasets is given. These approaches are comprehensively compared to bring out a clear differentiation among them. This article also surveys the problem of high-dimensional data and the existing approaches, that makes it more relevant. It also explores the latest trends in cluster analysis, and the real-life applications of this concept. This survey is exhaustive as it tries to cover all the aspects of clustering in the field of data mining.

CCS Concepts: • **Computing methodologies → Cluster analysis**;

Additional Key Words and Phrases: Cluster analysis, dimensionality reduction, data clustering applications, clustering tendency, large scale data mining, data clustering process

ACM Reference format:
Divya Pandove, Shivani Goel, and Rinkle Rani. 2018. Systematic Review of Clustering High-Dimensional and Large Datasets. *ACM Trans. Knowl. Discov. Data.* 12, 2, Article 16 (January 2018), 68 pages.
https://doi.org/10.1145/3132088

1 INTRODUCTION

Data clustering is a very important form of knowledge discovery tool. There is a plethora of approaches that deal with the implementation of the data clustering process. To understand

Authors' addresses: D. Pandove (Corresponding author) and R. Rani, Computer Science and Engineering Department, Thapar University, Patiala, Punjab 147004, India; emails: dpandove@gmail.com, raggarwal@thapar.edu; S. Goel, Department of Computer Science Engineering, School of Engineering and Applied Sciences, Bennett University, Greater Noida, U.P., India; email: shigo108@yahoo.com.

https://doi.org/10.1145/3132088

different facets of clustering, there is a need for extensive literature survey on this topic. The survey presented in this article focuses on understanding clustering, both as a mathematical problem and as a process. The thumb rule on which this survey has evolved is that only good data can produce good clusters. The goodness of data is directly proportional to the fulfillment of its purpose. By good data we mean, if we extract the right data corresponding to a particular problem, we have good data at hand. This brings us down to finding the most effective data mining techniques that would give us good data. Hence, data mining methodologies applied in clustering need to be properly enlisted and understood. In this article, we understand the concept of data clustering from both computational and mathematical perspectives. We then present a short summary and comparative analysis of the similar surveys performed in the past. We further our understanding of data clustering by understanding the clustering process for data mining. This includes understanding concepts such as clustering tendency, feature selection, feature extraction, cluster validation (such as internal indices, external indices), and relative measures. The most important contribution of this article is a collection of approaches that deal with large and high-dimensional datasets. The discussion on clustering large-scale datasets includes high-dimensional enlisting distance and similarity measures, that are important parameters for performing data clustering. Approaches for big data mining are extensively covered and compared. The comparison, presented in Table 3, gives a clear view to the readers about the traditional and recent clustering algorithms, used to perform data clustering on large datasets. We have also discussed the problem of cluster formulation in high-dimensional data, and the clustering approaches used to cluster such data. To make this review more relevant to the present time, challenges, and latest trends in large scale, and high-dimensional data clustering are also presented. This shows the direction in which data clustering is progressing and the prospects of research. Applications of the clustering approaches, in real time, are also presented along with a mapping of clustering approaches to applications.

1.1 An Introduction to Clustering

The term data clustering came into print for the first time in the year 1954, when an article dealing with anthropological data had this term in its title [30]. Cluster analysis has its origin in domains, such as machine learning, artificial intelligence, data mining, biology, statistics, and so on. Different fields use distinctive names for cluster analysis. Some of them are as follows: Q Analysis, data visualization, typology, numerical taxonomy, clumping, and so on [125, 152]. Clustering is a familiar form of data mining for the classification of data objects. This classification is based on similarity and dissimilarity among data objects. In data mining, data objects are represented as points. The point to note is that the classification of data into clusters is unsupervised and can be denoted as $X = C_1 \cup ....C_i \cup C_n; C_i \cap C_j = \Phi (i \neq j)$. Here, X denotes original dataset, C_i, C_j are clusters formed in X and n denotes the number of clusters formed [93]. Unsupervised knowledge means that one can have former information on the domain of the data but not of the structural characteristics. The most commonly unknown characteristic is the spatial distribution of the data. It represents information about a cluster such as its volume, density, shape, or orientation [68].

1.1.1 Clustering as a Computational Problem. Clustering is a common form of data mining technique having applications in many fields such as data compression, computer vision, texture segmentation, and vector quantization. Most of the clustering methods become computationally intractable with increase in the problem scale, as these methods have combinatorial nature [30]. For some specific objective functions, if the number of clusters exceeds three, clustering becomes a NP hard problem. This may be explained as when a program based on an algorithm is devised and run, it may deliver optimal results when tested with test examples, but when using real-time datasets, the test results may vary from the expected results. This may be because of an inefficient algorithm, or a complex computational problem, or a combination of the two [35]. While solving the cluster analysis problem, we take original set of data points and partition them to form k groups. These groups are mutually exclusive and exhaustive in nature. Most common criterion to determine the heterogeneity of a given cluster is the comparison of distances. The three possible conditions are:

(1) Define the maximum cluster distance and minimize it
(2) Calculate the sum of averages of the cluster distances and minimize it
(3) Calculate the total cluster distance and minimize it

When we take possible number of clusters as "K," the computational analysis of all partitions demonstrated by all the algorithms give good solutions, but they may or may not be optimal solutions [178].

1.1.2 A Mathematical Theorem for Clustering. Let there be a partition P_k, that divides the data space into K clusters. It is assumed that it is a perfect partition. It is given that all the distances among data points of a cluster are zero. There is no condition on the distance between the clusters. Hence, the clusters have to be separated well. Let $d(P_k)$, be the class of the criteria "minimize," where $d(P_k)$ is a function that calculates distances within a cluster, such that $d(P_k) \geq 0$ and $d(P_k) = 0$, if and only if, (P_k) is a perfect partition. This can be written in the form of a theorem: Optimal clustering is NP-Hard for any criterion minimize $d(P_k)$, such that $d(P_k) \geq 0$ and $d(P_k) = 0$ if and only if P_k is a perfect partition [61].

1.2 Related Research

Data clustering is an evolving field, many researchers have contributed to the literature in this area. In this section, we present a summary of the contributions made by other researchers in this area. The work in [181] is one of the earlier contributions in survey of clustering algorithms. The authors have surveyed clustering algorithms used in different domains and their applications on benchmark datasets, and computational problems. They have also discussed many closely correlated topics such as cluster validation, and proximity measures. The authors in [41] have talked about applications, challenges, techniques, and technologies on big data. They have tried to present a list of technologies used these days to deal with big data. Their focus is on clustering techniques based on MapReduce and parallel classification using MapReduce. The authors in [81] have given a review on rise of big data on cloud computing and have addressed open research issues. The authors in [66] have used taxonomy and empirical analysis, to survey clustering algorithms on big data. They have presented a survey of different categories of existing clustering algorithms (partitioning-based, model-based, grid-based, hierarchical-based, and density-based). The authors have given a comparison between these five categories of data clustering. Their goal is to find the best performing algorithm for big data clustering. In [20], the authors have presented the most popular algorithms in literature and a comparison between them. The authors have also presented ideas to build a clustering technique based on fuzzy clustering methods. The authors in [42] have surveyed big data from data processing, and usage perspective. This research barely touches data clustering but presents a whole new perspective to survey big data. The authors in [14] have focused only on the clustering algorithms on density based data stream clustering. Clustering of data streams is emerging as an important field of research. The work in [17] is a survey of clustering techniques for big data analysis. The authors have used comprehensive comparison to find the most appropriate big data clustering algorithm. The authors in [186] define big data and list the challenges presented by data of such huge magnitude. They have reviewed both single and multiple machine clustering techniques. Some of the work corresponds to clustering algorithms dealing with some specific domains such as machine learning, ontology, data mining, information extraction, text clustering, pattern recognition, bio-informatics, mobile ad-hoc networks, and so on, [10, 31, 33, 185]. Table 1 presents a comparison of existing surveys on clustering large-scale and high-dimensional data.

1.3 Our Contribution

We have presented the problem of clustering in a comprehensive manner, addressing the gaps left by other surveys. We have not only talked about the approaches dealing with big and high-dimensional data clustering, but have also discussed the building blocks of data clustering. This along with latest trends in clustering, real-time applications in this field and challenges faced, make this manuscript a complete guide on clustering large scale and high-dimensional datasets.

The rest of the article is organized as follows: Section 2 elaborates the process of clustering in data mining. It gives a detailed account of the process based on data treating and also gives a macroview of the cluster analysis by enlisting the steps involved in the process. These steps involve feature selection and extraction, cluster algorithm design, cluster validation, and result interpretation (external validation, internal validation, and relative measures). Section 3 discuses clustering of large scale data. It contains a list of distance and similarity measures, along with big data clustering approaches, such as hierarchical clustering, *k*-means clustering, random sampling, randomized search approach, condensation-based approach, density-based approach, grid-based approach, probabilistic model-based approach, and graph and network data. This section also briefly discusses the evolving role of Apache Hadoop in the field of big data. Section 4 contains information on clustering high-dimensional datasets. It discusses the problem of cluster formulation in high-dimensional data and clustering approaches dealing with this problem. Section 5 embodies challenges and latest trends in large-scale and high-dimensional data clustering. Section 6 enlists real-time applications of large-scale and high-dimensional data clustering. Section 7 gives the conclusion of this exhaustive survey and proposes the future scope. The outline of this survey is given in Figure 1.

2 CLUSTERING PROCESS FOR DATA MINING

In this section, we discuss concepts and process that help us define data mining using clustering.

2.1 Clustering Tendency

Data must be tested to see if it has some inherent grouping structure, to determine clustering tendency or clusterability of data. It is a hard task as there are so many different definitions of clusters. There are some clusterability assessment methods that can be used for this purpose, discussed as follows [148]:

(1) Spatial Histogram: In this approach, the histogram generated from random samples is contrasted with a d-dimensional histogram generated from the input dataset D. dataset D, is clusterable if the distributions of the histograms are rather different. Each dimension is divided into qui-width bins, and the number of points in each cell is counted. Then, the empirical joint probability mass function (EPMF) is calculated. This is done for the randomly sampled data as well. The difference in the two is calculated by using the Kullback–Leibler (KL) divergence value. If the two values are different then we say that the dataset is clusterable [69]. Figure 2 shows the contrast in histograms produced from randomly generated data samples and the one produced from a dataset.
(2) Distance distribution: Compare the pairwise point distance from the data with those from randomly generated samples.
(3) Hopkins statistic: The spatial randomness is tested by using this sparse sampling test.

Table 1. Comparative Study of the Existing Research/Surveys

Research paper	Domain	Defines clustering process	Clustering big data	Preprocessing of data	High-dimensional data clustering	Trends in cluster analysis	Applications of cluster analysis	Challenges faced
Survey of clustering algorithms [181]	Machine learning, statistics, Computer science	✓	✓		✓		✓	
A survey of ontology evaluation techniques [33]	Ontology evolution						✓	
A survey on clustering algorithms for wireless sensor networks [31]	Wireless sensor networks						✓	✓
A survey of text clustering algorithms [10]	Text clustering					✓	✓	✓
Data-intensive applications, challenges, techniques and technologies: A survey on Big Data [41]	Granular computing, cloud computing, bio-inspired computing, quantum computing						✓	✓
The rise of big data on cloud computing: Review and open research issues [81]	Cloud computing							✓
A survey of clustering algorithms for big data: Taxonomy and empirical analysis [66]	Machine learning, statistics, Computer science		✓		✓	✓		
Big data: A survey [42]	Cloud computing, internet of things			✓	✓		✓	✓
On density-based data streams clustering algorithms: a survey [14]	Data stream		✓		✓	✓		✓
Big data clustering: Algorithms and challenges [186]	Health, bio-medical, marketing, finance, marketing, transport		✓		✓			✓
Survey on clustering methods: Towards fuzzy clustering for big data [20]	Machine learning, statistics, Computer science		✓		✓		✓	
A survey of clustering techniques for big data analysis [17]	Machine learning, statistics, Computer science		✓		✓			

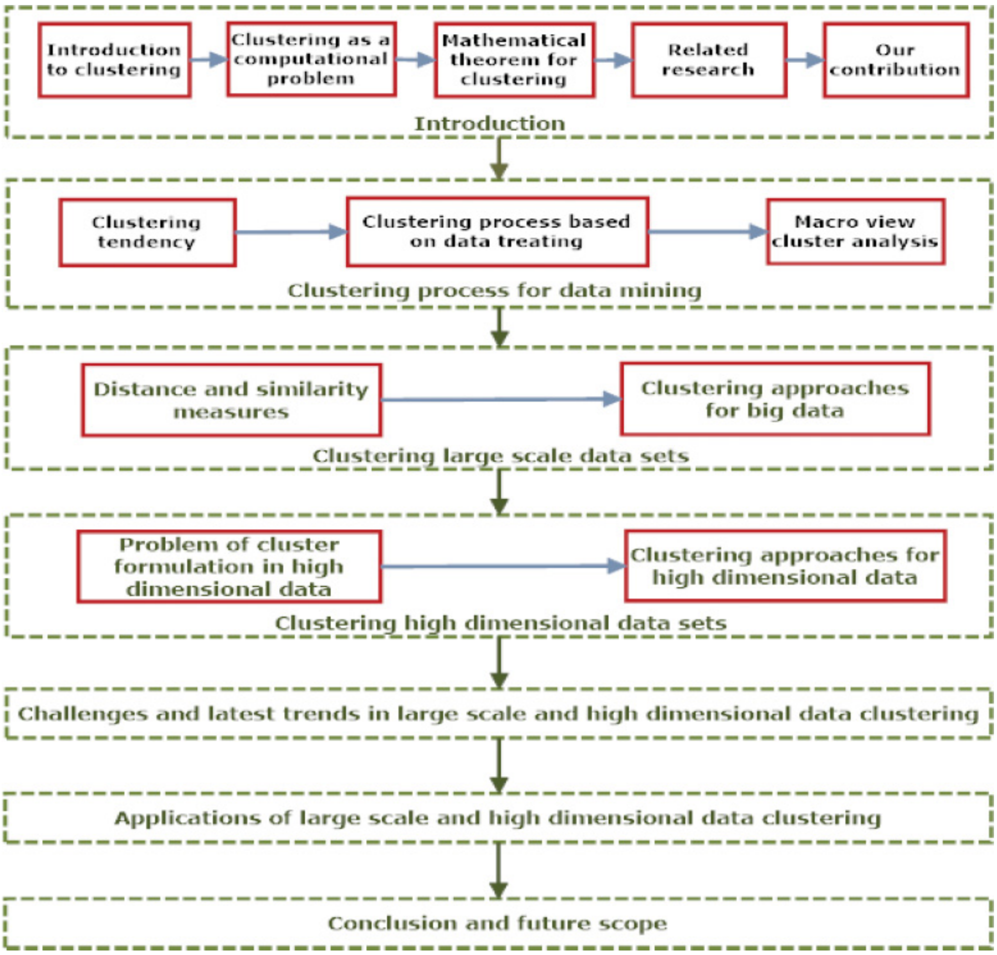

Introduction to clustering
Clustering as a computational problem
Mathematical theorem for clustering
Related research
Our contribution
Introduction
Clustering tendency
Clustering process based on data treating
Macro view cluster analysis
Clustering process for data mining
Distance and similarity measures
Clustering approaches for big data
Clustering large scale data sets
Problem of cluster formulation in high dimensional data
Clustering approaches for high dimensional data
Clustering high dimensional data sets
Challenges and latest trends in large scale and high dimensional data clustering
Applications of large scale and high dimensional data clustering
Conclusion and future scope

Fig. 1. Layout of the manuscript.

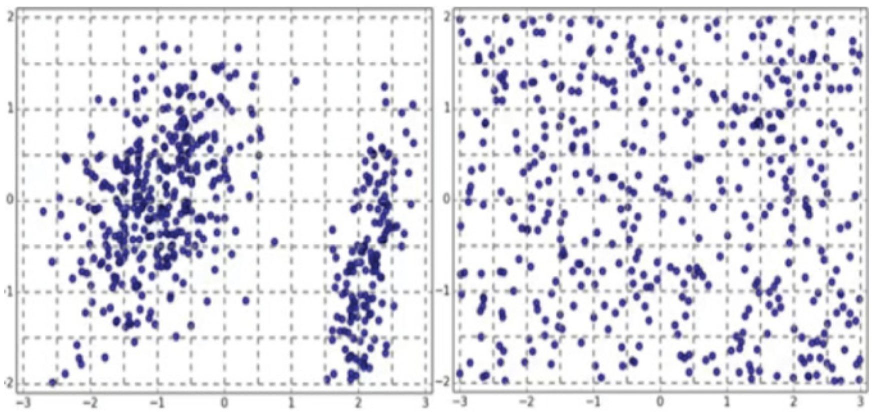


Fig. 2. Histograms generated by using random data samples and a real dataset.

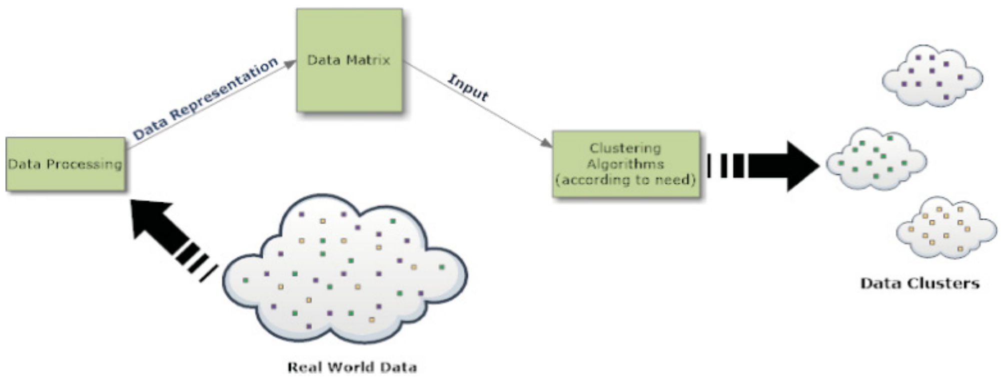


Fig. 3. Clustering process based on data processing.

The process of clustering for data mining can be understood by two complementary perspectives. First, from the point of view of data treatment (microview) that deals with cleaning, integration, reduction, interpretation, and transformation of data points. Second, the macroscopic view of the clustering process, which includes feature extraction, algorithmic design, cluster validation, and result interpretation. Both these perspectives highlight different views of cluster formulation with each having unique positives, and some inevitable fallacies. The clustering algorithms or approaches will inevitably produce clusters, the only concern is cluster quality. The resultant cluster evaluation cannot be solely based on the clustering technique applied, it also depends on the data domain. Both these processes lay huge emphasis on data processing.

2.2 Clustering Process Based on Data Treating

Any data in its initial, real world state is raw data and needs to be processed to prepare it for processing procedure of any kind. This is known as data pre-processing, which includes cleaning of data, its integration, reduction, and transformation. Data Cleaning involves identifying and assigning missing values, clearing the noise in the data, outlier identification and separation, and addressing the inconsistencies in data. Data Integration is the process of integrating multiple files, databases, or data cubes. Data Reduction results in decrease in the volume of data, but without any change in the analytical results. Data transformation focuses on normalization and aggregation of data [150]. Once the data is processed, it is represented in the form of data matrices. This is done in order to provide input to different clustering algorithms, selected according to need and resource availability. The final output of the entire process is data clusters of different shapes (according to distribution of data) and sizes [70]. Given the voluminous increase in data, it has become very important to extract meaningful information with visible correlations. Clustering as a concept assumes that relevant data should be processed in order to form meaningful and syntactically correct clusters. The process of data treating aims to give relevant and well presented data as an input to the selected clustering algorithms. This ensures that the cohesion in data of the resultant clusters is maximized. The process is visualized in Figure 3.

2.3 The Macroview Cluster Analysis

A typical clustering process has four major steps. All these steps are related to each other and have a feedback path. The clusters obtained at the end of these steps are an accumulated result of these steps.

2.3.1 Feature Selection and Extraction. Feature selection is used to identify unique features, that can form a subset that can later be used to define a cohesive cluster. Feature extraction selects differentiating features from a set of chosen members and uses a transforming function to generate some new salient features. This is the most important step in cluster processing, as it increases the effectiveness of cluster applications. If feature selection is effective, then the effort in the design phase is greatly simplified. The entire clustering process is based on this operation as incorrect features selected may result in an increase in the complexity and can also result in the creation of irrelevant clusters [27, 92]. The criteria for selection of features may be qualitative or quantitative in nature, depending on whether they can be measured or not. Quantitative features can be measured using different scales such as ratio, ordinate, or nominal. Qualitative features are more abstract in nature and are difficult to represent [74]. Structured features, that are represented in the form of trees, can also be used. In this case, the parent node represents generalization and the child node acts as its specification [123].

It might seem that feature selection is similar to data reduction, discussed in the previous subsection (clustering process based on data processing). Though they might seem similar, but there are basic differences that set these two concepts apart. Data reduction techniques combine similar (correlated) attributes and create new ones. The new attributes are superior to the original components. Feature selection, on the other hand, does not combine attributes, just evaluates their quality, predictive power, and selects the best set. There are many specific data reduction methods like PCA (Principle Component Analysis), but feature selection does not refer to a specific technique or to a specific objective function. The result of feature selection is a subset of the original data set, while data reduction techniques output a transformed dataset. Both the techniques can be combined by creating reduced component set, and then running feature selection on them.

2.3.2 Clustering Algorithm Design or Selection. If we try to formulate a single clustering algorithm, that satisfies the needs of all the real world problems, then we would fail miserably. This is stated in the *insolvability theorem* as "No single clustering algorithm simultaneously satisfies the three basic axioms of data clustering, i.e., scale-invariance, consistency and richness" [100]. This means that it is impossible to develop a global model of clustering approaches that applies to all the different fields. The algorithm needs to be selected very carefully keeping domain knowledge in mind. Mostly all the algorithms are directly or indirectly connected to the proximity measure. They are generally based on input criteria such as number of clusters, termination condition, optimization criterion, and so on. Proximity measures form the most important members of this group. Hence, all these things are to be kept in mind before selecting the algorithm for a given application [182].

2.3.3 Cluster Validation. It has been observed that different clustering algorithms tend to produce different results when applied to the same datasets. Even when a given algorithm is altered by changing its parameters, it tends to produce different results. This goes on to prove the *impossibility theorem.* Another issue with clustering validation is that, being an unsupervised method, there is no expert to judge goodness of cluster. While evaluating a cluster the goodness of the clustering is measured. It can be done by evaluating the clustering stability for understanding the clustering result's sensitivity to different parameters of algorithm, i.e., number of clusters, and so on. Clustering tendency is another factor for assessing the cluster suitability, i.e., whether data has any inherent grouping structure or not. To validate the clusters produced by the clustering algorithms, we broadly consider three software engineering criteria [58]:

(1) Internal indices: The clusters obtained from a given clustering algorithm are compared with data only; it is an unsupervised criterion derived from data itself. Goodness of

Table 2. Variables Used to Define External Methods of Clustering Validation

Symbol	Meaning
n	Total number of data points
n_i	Maximum no. of points in a cluster
$n - j$	Maximum no. of ground truths in a cluster
n_{ij}	Distribution of ground truths in a cluster
i	Clusters
j	Ground truths
r	Desired no. of clusters
C_i	A cluster in C
k	No. of clusters
$w(e_{ij})$	Weight of each element in a cluster
$w(M)$	Sum of weights of all elements
n_{ij}	Number of clusters having ground truths
T_j	Partition based on ground truths
f_i	F-measure
PC_i	Probability of clustering C_i
PT_i	Probability of partitioning T_i
$H(T)$	Entropy of partitioning T
$H(T\|C_i)$	Entropy of partitioning T with respect to cluster C_i
$I(C,T)$	Mutual information
$NMI(C,T)$	Normalized mutual information
x_i, y_i	Data points

the clustering can be evaluated by understanding cluster separation and compactness. Silhouette Coefficient is a good example of this case.

(2) External indices: External knowledge is collected over time to provide a knowledge base. A comparison is made between this knowledge base and the clusters obtained from the application of the clustering algorithm. It is a supervised index and employs criteria not inherent to the dataset.

(3) Relative indices: A comparison of clusters produced by applying different clustering techniques on the same datasets. They are used to compare different clusterings, generally retrieved through distinct settings of parameters when algorithm considered is same. For example, different no. of clusters for same algorithm may generate different results.

Let us discuss various methods of validation for clustering measures [78]. Table 2 lists all the variables used to define the external methods for cluster validation.

(1) External validation for clustering measures
For implementation of this measure ground truths are used. Given ground truth T,Q(C,T), where Q(C,T) represents clustering C's quality measure. If Q(C,T) satisfies the following conditions, then it is good:
(a) *Homogeneity of cluster*, i.e., it is better if the cluster is pure.
(b) *Cluster completeness*: It means the objects in ground truth having relationships within the same category are assigned to the same cluster.

(c) *Rag bag better than alien*: Meaning that putting heterogenous objects into a rag bag (miscellaneous or other category) should be less penalized than putting the objects into a pure cluster.
(d) *Prevention of small cluster,* i.e., to split a large category into pieces is better than to split a small category into pieces.

The external measures that are most commonly used are as follows:

(a) Measures based on matching [154]: The most common matching based measures are discussed below:
   (i) Purity: This measure is based on an assumption that the cluster C_i contains points only from one (ground truth) partition. Mathematically, it can be represented as

$$purity_i = \frac{1}{n_i} max_{j=1}^{k}\{n_{ij}\}. \tag{1}$$

Total purity of the clustering C: Purity of each cluster is calculated and added to get the total purity.

$$purity = \sum_{i=1}^{n} \frac{n_i}{n} purity_i = \frac{1}{n}\sum_{i-1}^{r} max_{i=1}^{k}\{n_{ij}\}. \tag{2}$$

Perfect clustering is obtained if purity = 1 and $r = k$, i.e., the obtained number of clusters is same as that in the clusters in the ground truth.

   (ii) Maximum matching: This measure is based on the rule that only one cluster can match one partition. The matching done in this case is pairwise matching, this means that one element only belongs to one cluster. If weight of each point is given by $w(e_{ij}) = n_{ij}$, then maximum weight match is given by [13]

$$match = \arg\max_M \left\{\frac{w(M)}{n}\right\} \tag{3}$$

   (iii) F-Measure: In information retrieval, it is considered as a very popular measure, which is calculated by precision and recall.
      (A) Precision: It is defined as the fraction of points in C_i from the majority partition T_j (i.e., the same as purity), where j_i is the partition containing the maximal points from C_i. Mathematically it is represented as

$$prec_i = \frac{1}{n_i} max_{j=i}^{k}\{n_{ij}\} = \frac{n_{ij_i}}{n_i}. \tag{4}$$

      (B) Recall: The fraction of points present in partition T_{j_i} are shared commonly with cluster C_i, where $m_j =\mid T_j \mid$. It is represented as

$$recall_i = \frac{n_{ij_i}}{\mid T_{j_i} \mid} = \frac{n_{ij_i}}{m_{j_i}}. \tag{5}$$

F-measure for C_i is the harmonic means of $prec_i$ and $recall_i$ and is represented as

$$f_i = \frac{2n_{ij}}{n_i + m_j}. \tag{6}$$

The average of all clusters is considered as F-measure for clustering C and represented as

$$f = \frac{1}{r}\sum_{i=1}^{r} F_i. \tag{7}$$

(b) Entropy measures: Entropy is a very useful information theory measure. It is also used in machine learning and data mining quite a lot. It represents the amount of orderness of the information in all the partitions. Entropy for clustering C for r clusters is represented as [154]

$$H(C) = -\sum_{i=1}^{r} PC_i \log PC_i, PC_i = \frac{n_i}{n}. \tag{8}$$

Some entropy-based measures are

—Conditional entropy: If PC_i represents the probability of cluster C_i, n_i is the no. of points contained in a cluster and n is total no. of points. Entropy of partitioning T for ground truth j for k groups is represented as

$$H(T) = -\sum_{j=1}^{k} PT_i \log PT_j. \tag{9}$$

Entropy of T w.r.t. cluster C_i, i.e., how ground truths are distributed within each cluster is represented as

$$H(T \mid C_i) = -\sum_{j=1}^{r} \left(\frac{n_{ij}}{n_i}\right) \log \left(\frac{n_{ij}}{n_i}\right). \tag{10}$$

T's conditional entropy with respect to C clustering:

$$H(T \mid C) = -\sum_{i=1}^{r} \left(\frac{n_i}{n}\right) H(T \mid C_i) = -\sum_{i=1}^{r}\sum_{j=1}^{k} P_{ij} \log \frac{P_{ij}}{P_{C_i}}. \tag{11}$$

The conditional entropy is higher when more number of cluster members are being split into various partitions. For perfect clustering, the conditional entropy value is 0, the worst attainable value of conditional entropy is $\log K$.

$$\begin{aligned} H(T \mid C) &= -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij}(\log p_{ij} - \log PC_i) \\ &= -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log p_{ij} + \sum_{i=r}^{r}\left(\log PC_i \sum_{j=1}^{k} P_{ij}\right). \end{aligned} \tag{12}$$

Concisely the equation can be written as

$$-\sum_{i=1}^{r}\sum_{j=1}^{k} P_{ij} \log P_{ij} + \sum_{i=1}^{r}(PC_i \log PC_i) = H(C,T) - H(C). \tag{13}$$

—Normalization Mutual Information (NMI) [84]: If C is the clustering and T is the partitioning, then the quantity of information shared between C and T is quantified by mutual information. It is expressed as

$$I(C,T) = \sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log \left(\frac{p_{ij}}{PC_i.PT_j}\right). \tag{14}$$

This expression implies that dependency between the observed joint probability P_{ij}, C, T, and expected joint probability PC_i is measured. It is assumed that PT_j is independent. $P_{ij} = P_{ci} \cdot P_{Tj}$, $I(C,T) = 0$ if C and T are considered independent. There is no upper bound on mutual information, which is not desirable, and hence needs to be normalized. The normalized value of the mutual information lies in the range of [0,1], 1 indicating good clustering. It is represented as

$$NMI(C,T) = \sqrt{\frac{I(C,T)}{H(C)} . \frac{I(C,T)}{H(T)}} = \frac{I(C,T)}{\sqrt{H(C) \cdot H(T)}}. \tag{15}$$

(c) Pairwise measures: This measure has four possibilities based on the agreement between cluster label and partition label [80]:

(i) TP: True positive: Two points x_i and x_j exist in the identical partition T, and identical cluster C. It is represented as

$$TP = \mid \{(x_i, x_j) : y_i = y_j \; and \; \hat{y}_i = \hat{y}_j\} \mid, \tag{16}$$

where $\hat{y}_i$ is the cluster label for point x_i and y_i is the true partition label.

(ii) FN: False negative is represented as

$$FN = \mid \{(x_i, x_j) : y_i = y_j \; and \; \hat{y}_i \neq \hat{y}_i\} \mid . \tag{17}$$

(iii) FN: False positive is represented as

$$FN = \mid \{(x_i, x_j) : y_i \neq y_j \; and \; \hat{y}_i = \hat{y}_i\} \mid . \tag{18}$$

(iv) FN: True negative is represented as

$$FN = \mid \{(x_i, x_j) : y_i \neq y_j \; and \; \hat{y}_i \neq \hat{y}_i\} \mid . \tag{19}$$

The four measures are calculated as shown below:

(i) True positive:

$$TP = \sum_{i=1}^{r} \sum_{j=1}^{k} (2^{n_{ij}}) = \frac{1}{2}\left(\left(\sum_{i=1}^{r} \sum_{j=1}^{k} {n_{ij}}^2\right) - n\right).$$

(ii) False negative:

$$FN = \sum_{j=1}^{k} (2^{m_j}) - TP.$$

(iii) False positive:

$$FP = \sum_{i=1}^{r} (2^{n_i}) - TP.$$

(iv) True negative:

$$TN = \frac{1}{2}\left(n^2 - \sum_{i=1}^{r} n_i^2 - \sum_{j=1}^{k} m_j^2 + \sum_{i=1}^{r} \sum_{j=1}^{k} n_{ij}^2\right).$$

Other pairwise measures include the Jaccard Coefficient, Rand Statistic, and the Fowlkes Mallow measure [36]. The Jaccard Coefficient can be expressed in terms of true positive and negative as

$$Jaccard = TP/(TP + FN + FP). \tag{20}$$

This means that denominator ignores TN. It has value 1 for perfect clustering.

Rand Statistic is given by

$$Rand = (TP + TN)/N. \tag{21}$$

The value of Rand Static for a symmetric and perfect clustering is 1.

Fowles–Mallow measure: It can be defined as geometric mean of precision and recall and can be expressed as

$$FM = \sqrt{prec\ X\ recall} = \frac{TP}{\sqrt{(TP + FN)(TP + FP)}}. \tag{22}$$

(2) Internal measures: In many cases ground truths are not available, that is when we have to rely on internal measures. The internal measures are based on the concept of clustering meaning that the points have to be present within the same cluster. A trade off between maximizing intra-cluster compactness and inter-cluster separation is required. Given a clustering $C = \{C_1, \ldots, C_k\}$ with k clusters, cluster C_i contains $n_i = \mid C_i \mid$ points. Let $W(S, R)$ be the summation of weights on all edges having one vertex in cluster S and other vertex in cluster R. The summation of overall intra-cluster weights over entire clusters:

$$W_{in} = \frac{1}{2} \sum_{i=1}^{k} W(C_i, C_i). \tag{23}$$

Here, (C_i, C_i) means that both vertices are in the same cluster. Weight (W) can be defined as a distance measure. The summation of all the inter-cluster weights are represented as

$$W_{out} = \frac{1}{2} \sum_{i=1}^{k} W(C_i, \overline{C_i}) = \sum_{i=1}^{k-1} \sum_{j>1} W(C_i, C_j). \tag{24}$$

Here, $(C_i, \overline{C_i})$ means that one vertex is in C_i and the other vertex is not in C_i. The number of distinct intra-cluster edges is given by

$$N_{in} = \sum_{i=1}^{k} (2^{n_i}). \tag{25}$$

The number of distinct inter cluster edges is given by

$$N_{out} = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} n_i n_j. \tag{26}$$

Here, one data point is in cluster (i) and the other is in ground truth (j). Some of the common internal measures are given below:

(a) Beta-CV Measure [147]: This measure expresses the ratio of the average intra-cluster distance to the average inter-cluster distance. The smaller the clustering, the better it is. It is expressed as

$$Beta\ CV = \frac{W_{in}/N_{in}}{W_{out}/N_{out}}. \tag{27}$$

(b) Normalized cut [36]: If a graph is to be partitioned into two sets X and Y such that the weight of edges connecting vertices in X to vertices in Y is minimum and size of X and Y is very similar then,

$$cut(X, Y) := \sum_{i \epsilon A, j \epsilon B} W_{ij},$$

where W_{ij} control size of the neighborhood, and is equal to $e^{\frac{|x_i - x_j|^2}{2\sigma^2}}$. Then, the normalized cut is given by

$$N_{cut}(A, B) := cut(A, B)\left(\frac{1}{vol(A)} + \frac{1}{vol(B)}\right), \tag{28}$$

where $vol(A) = \sum_{i\epsilon A} d_i$.

The higher the normalized cut value, the better the clustering.

(c) Modularity [164]: This measure is used for graph clustering and can be defined as

$$Q = \sum_{i=1}^{k}\left(\frac{W(C_i, C_i)}{W(V, V)} - \left(\frac{W(C_i, V)}{W(V, V)}\right)^2\right). \tag{29}$$

Here, $\frac{W(C_i,C_i)}{W(V,V)}$ represents observed fraction of weights, $(\frac{W(C_i,V)}{W(V,V)})$ represents expected fraction of weights, and Q represents modularity:

$$W(V, V) = \sum_{i=1}^{k} W(C_i, V) = \sum_{i=1}^{k} W(C_i, C_i) + \sum_{i=1}^{k} W(C_i, \overline{C_i}) = 2(W_{in} + W_{out}). \tag{30}$$

Modularity measures the difference between the observed and expected fraction of weights on edges within the clusters. The smaller the value, the better the clustering and the intra-cluster distances are lower than expected.

(3) Relative measures: These measures directly compare disparate clusterings, generally that are attained by changing different parameter settings for the similar algorithm. We discuss the Silhouette Coefficient that can act as both internal or relative measure. Both these cases are discussed below [16]:

(a) Silhouette Coefficient as internal measure: It checks cluster cohesion and separation. For each point x_i, its Silhouette Coefficient s_i is defined as

$$s_i = \frac{\mu_{out}^{min}(x_i) - \mu_{in}(x_i)}{max\{\mu_{out}^{min}(x_i), \mu_{in}(x_i)\}}, \tag{31}$$

where $\mu_{in}(x_i)$ is the mean distance measure from x_i to points in its closest cluster and μ_{out}^{min} is the mean distance measure from x_i to points in its closest cluster.

We can say that Silhouette Coefficient is the mean value of s_i across all the points. This statement can be represented as

$$SC = \frac{1}{n}\sum_{i=1}^{n} s_i. \tag{32}$$

SC like all other coefficients also implies good clustering when it is closer to +1. This means that points are closer to their own clusters but are far from other clusters.

(b) Silhouette coefficient as relative measure: This measure estimates the count of clusters in the data in this case. A value of k is selected that yields the best clustering which means that it yields high values of SC and SC_i (SC as relative measure and $1 \leq j \leq k$). The equation for the same is given as

$$SC_i = \frac{1}{n}\sum_{x_j \in c_i} S_j. \tag{33}$$

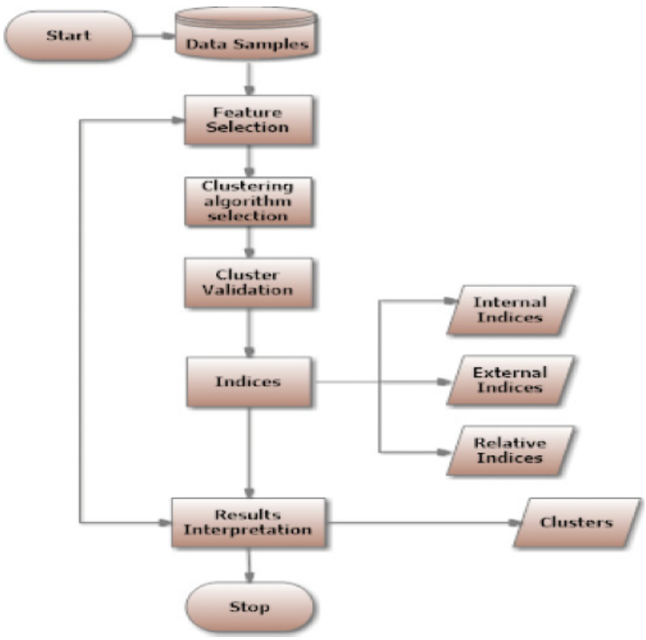


Fig. 4. Flow of the general clustering process.

2.3.4 Results Interpretation. This is the final step of the clustering process [182]. The final aim of the process of clustering it to provide data handlers with significant insights into seemingly meaningless data. This is done for increasing the efficiency of data analysis. This area has huge research potential and nowadays, huge emphasis is given on devising effective result interpretation mechanisms. The flow of these steps and the feedback mechanism of the process is illustrated in Figure 4.

3 CLUSTERING LARGE-SCALE DATASETS

With the evolution of technology, the volume of data collected in any and all domains is multiplying. The definition of large data is very subjective as there is no specific data size above which data can be classified as large. The parameters for such classification are also not set in stone. Large datasets can be defined in terms of volume, processing time, pattern approximation, and so on, depending on the application and evaluation criterion. As the volume of incoming data keeps on increasing, the challenges of handling this data are also increasing exponentially. The increase in complexity of data mainly results into two major challenges:

(1) Developing approaches to handle massive datasets.
(2) Tackling the problem of high-dimensionality in datasets.

In this section, we discuss a major challenge of increasing data complexity and scalability i.e., handling massive datasets. We concentrate on discussing an important solution to deal with such datasets—data clustering. In order to understand the concept of data clustering better, we define distance and similarity measures used by many of the major algorithms. Then, we enlist major algorithms that are used to deal with large/big datasets.

3.1 Distance and Similarity Measures

Data point is an abstract term, that can be used to describe a particular feature, or members of a set of features. It can be qualitative or quantitative in nature. After determining the nature of data points, other mechanisms to be used for their evaluation are decided. The comparison of two data points can be done using either the measures of similarity or those of distance. These two measures can be explained as

(1) Similarity measures: These can be defined as the parameter, that evaluates the closeness between a pair of data points; a data point and a cluster or two different clusters. It can be understood using Jaccard Similarity (JS) of sets, represented as $|A \cap B|/|A \cup B|$, i.e., JS is the ratio of the intersection of A and B to the their union. Similarity measures can be applied in many domains such as finding similarity in documents (used in applications such as detecting plagiarism, finding mirror pages, or articles from the same source, etc.), collaborative filtering (used in developing recommendation systems), prediction of online purchases, movie ratings, consumer behavior, and so on [151]. If we consider a dataset X, then the similarity function F satisfies the following conditions:
   (a) $F(x_i, x_j) = F(x_j, x_i)$
   (b) $0 \leq F(x_i, x_j) \leq 1$
   $(for\ all\ x_i\ and\ x_j)$
   (c) $F(x_i, x_j)F(x_j, x_k) \leq [F(x_i, x_j) + F(x_j, x_k)]F(x_i, x_k) for\ all\ x_i\ , x_j\ and\ x_k$
   (d) If $F(x_i, x_j) = 1\ and\ x_i = x_j$. (It is called a similarity matrix.)
   A dataset with N input patterns can form a proximity matrix to from a symmetric matrix of $N \times N$ order. The (i, j)th element of this matrix will represent the similarity or dissimilarity measure for ith and jth patterns. The datasets will be represented by columns of the matrix and the corresponding elements by rows.
(2) Dissimilarity or distance measures: If we consider a dataset X, then the dissimilarity function D has to satisfy the following conditions [182]:
   (a) $D(x_i, x_j) = D(x_j, x_i)$
   (b) $D(x_i, x_j) \geq 0$
   $(for\ all\ x_i\ and\ x_j)$
   (c) $D(x_i, x_j) \leq D(x_j, x_k) + D(x_k, x_j)$
   $(for\ all\ x_i,\ x_j\ and\ x_k)$
   (d) $D(x_i, x_j) = 0\ if\ x_i = x_j$.
   Generally, the quantitative features are measured by distance functions while, qualitative variables are more aptly given by similarity measures. There is no hard line separating the two. Some of the similarity and dissimilarity measures for quantitative features [182] are
   (a) Minkowski distance (L_p metric): In this case, the distance between two vectors is the absolute value of their difference. Larger valued features, or those with higher variance, usually dominate over other features [82]. If D_{ij} represents Minkowski distance between the data points x_i and y_i, it can be expressed as

$$D(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i|^n \right)^{\frac{1}{n}}. \tag{34}$$

   There are three special cases of Minkowski distance:
   (i) Euclidean metric: This is the most common form of metric. It can be taken as a special case of Minkowski equation with a power of 2. It tends to form

hyper-spherical clusters. It can be represented as [118]

$$D(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i|^2 \right)^{\frac{1}{2}}. \tag{35}$$

(ii) City block distance (Manhattan distance): It is a Minkowski equation with n as 1, with hyper-rectangular clusters and can be represented as [37]

$$D(x, y) = \sum_{i=1}^{d} |x_i - y_i|. \tag{36}$$

(iii) SUP distance: This is Minkowski equation with $n = \infty$, and are used in SUP norms with fuzzy c-means. It is represented as [28]

$$D(x, y) = \max_{1 \le l \le d} |x_i - y_i|. \tag{37}$$

(b) Mahalanobis's distance: It is used to measure the number of standard deviations a point P is away from the distribution D's mean. It is a dissimilarity measure between two randomly selected vectors, having same distribution with covariance matrix S. It corresponds to standard Euclidean distance, if the component principal axes is re-scaled to have unit variance (i.e., when there are no correlations). It is invariant to any non-singular linear transformation, and usually forms hyper-ellipsoidal clusters. It is within-group covariance matrix and is defined as [120]

$$D(x, y) = \sqrt{(x_i - y_i)^T S^{-1} (x_i - y_i)}. \tag{38}$$

There are other measures, which are not metrics but have their utility:

(c) Pearson's correlation: It is derived from the coefficient of correlation, and has the ability to measure the magnitude of difference between two variables. It gives a value between +1 and −1, to express the degree of correlation. It is widely used in medical studies to analyze genes expression. It can be expressed as [62]

$$D(x, y) = \frac{(1 - r)}{2}, \tag{39}$$

where r represents Pearson's correlation coefficient between data points x and y, and is given as

$$r = \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \overline{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \overline{y})^2}}. \tag{40}$$

(d) Point symmetry distance: It measures the distance between a reference point and an object. If a pattern is symmetric, the distance is minimized. It is usually applied in SBKM (Symmetry based K-means) [165].

$$D_{ir} = \min_{f=1,\ldots,N \, and \, j \neq i} \frac{||(x_i - x_r) + (x_j - x_r)||}{||(x_i - x_r)|| + ||x_j - x_r)||}. \tag{41}$$

(e) Cosine similarity: This measure is not dependent on vector length, and it varies with linear transformations, but not with rotation. It has wide application in document clustering. Similarity can be expressed as cos $\propto$ and is calculated as the dot product of components of data point vectors X and Y [162]:

$$S(X, Y) = \cos \propto = \frac{X.Y}{||X|| ||Y||}. \tag{42}$$

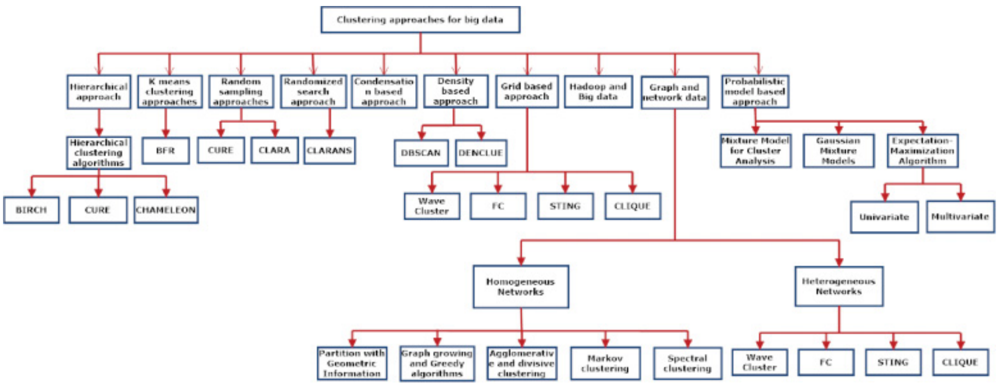

Clustering approaches for big data
Hierarchical approach
K means clustering approaches
Random sampling approaches
Randomized search approach
Condensation based approach
Density based approach
Grid based approach
Hadoop and Big data
Graph and network data
Probabilistic model based approach
Hierarchical clustering algorithms
BFR
CURE
CLARA
CLARANS
DBSCAN
DENCLUE
Mixture Model for Cluster Analysis
Gaussian Mixture Models
Expectation-Maximization Algorithm
BIRCH
CURE
CHAMELEON
Wave Cluster
FC
STING
CLIQUE
Univariate
Multivariate
Homogeneous Networks
Heterogeneous Networks
Partition with Geometric Information
Graph growing and Greedy algorithms
Agglomerative and divisive clustering
Markov clustering
Spectral clustering
Wave Cluster
FC
STING
CLIQUE

Fig. 5. Clustering approaches for big data.

This expression can be further expressed in terms of components of vector X and Y respectively as

$$\frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{1=1}^{n} Y_i^2}}. \tag{43}$$

3.2 Clustering Approaches for Big Data Mining

Once the data mining problem is stated clearly and it is clear that data clustering is the best machine learning approach to solve it, the next step is to select the clustering algorithm that will provide an optimal solution for the stated problem. There are innumerable clustering approaches available, but the scope of this manuscript extends to enlisting and comparing the approaches best suitable for large datasets. We also enlist some of the classical approaches for data clustering that are not in themselves equipped in dealing with massive dataset problems, but do form the basis for many other hybrid approaches. Clustering approaches discussed in this section are laid out in Figure 5.

3.2.1 Hierarchical Approach. This approach is best suited for clustering problems involving relatively small datasets. The basic principle involves checking the closeness of data points. The definition of closeness is subjective and can be picked up from many definitions of similarity and distance measures. The minimum distance can be calculated using the function:

$F(x_i, x_j) = min\ F(x_m, x_L)$ given that $1 \leq m, L \leq N$ (initial number of clusters) and $m \neq L$.

In the beginning, each point forms its own cluster that is why this type of clustering is also known as agglomerative clustering. Then, these clusters are iteratively merged depending on the chosen definition of closeness. The merger of clusters ends based on a particular criterion defined beforehand. The last iteration gives distinctively defined clusters. This algorithm is of cubic time complexity. The initial step takes $O(n^2)$ time. The following steps take $(n-1)^2$, $(n-2)^2$ time. The summation of squares is $O(n^3)$ making it cubic [48]. The result of a hierarchical algorithm is usually described in the form of a dendrogram, where a dataset is represented by its root and data objects by each leaf node. The distance measure between the objects as well as the clusters is represented by the height of the dendrogram. In order to visualize the real hierarchical relationships in data, we can cut the dendrogram at different levels [91].

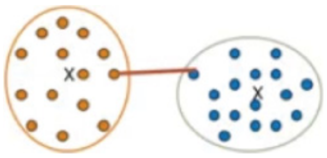


Fig. 6. Single linkage.

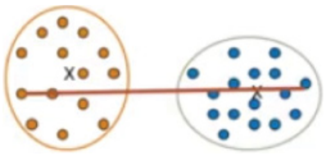


Fig. 7. Complete linkage.

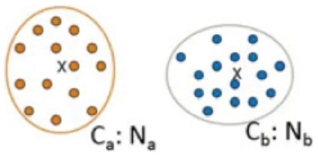


Fig. 8. Average linkage.

Based on the distance measure, there are mainly three agglomerative clustering approaches:

(1) Single linkage: Checks the closest distance amidst the two objects in different clusters, also known as the nearest neighbor search [160]. Figure 6 shows single linkage between two clusters. This kind of search focuses more on the close regions and ignores the overall structure of the cluster. It is capable of clustering non-elliptical shaped group of objects and is sensitive to outliers and noise.
(2) Complete linkage: Checks the farthest distance measure of two objects to determine the distance between any two clusters [161]. Figure 7 represents complete linkage between two clusters. The similarity between the two most dissimilar members is used to indicate the similarity between the two clusters. It merges two clusters to form one with the smallest diameter and is non-local in behavior. It helps in obtaining compact shaped clusters, and is sensitive to outliers.
(3) Average linkage: It is a more complicated agglomerative approach, which considers the mean distance between two objects belonging to different clusters [128]. Figure 8 shows average linkage between one element in first cluster and another element in second cluster. C_a and C_b represent two clusters. N_a and N_b represent the count of elements in clusters C_a and C_b, respectively. This measure is expensive to compute. Recently, a new algorithm Twister ties [47] has been proposed that uses local sensitivity hashing combined with a novel data structure called twister tries, to provide approximate clustering for average linkage. This approach only requires linear space. Further, the time complexity is linear in the number of items to be clustered, making it feasible to be applied to larger scale of data.

Both single and complete linkage can be represented by a recurrence formula [106]:

$D(C_l,(C_i,C_j)) = \alpha_i D(C_l,C_i) + \alpha_j D(C_l,C_j) + \beta D(C_i,C_j) + \gamma|D(C_l,C_i) - D(C_l,C_j)|.$

Here, D (**) is the distance function, α_i; α_j; β; and γ are coefficients, that take value according to the scheme used.

For Single Linkage the conditions are

$\alpha_i = \alpha_j = 1/2$, $\beta = 0$, and $\gamma = -1/2$, hence formula reduces to $D(C_l,(C_i,C_j)) = \min (D(C_l,C_i), D(C_l,C_j))$.

For Complete Linkage the conditions are

$\alpha_i = \alpha_j = \gamma = 1/2, \beta = 0$, hence the derived formula is $D(C_l,(C_i,C_j)) = max(D(C_l,C_i), D(C_l,C_j))$.

This particular approach is not appropriate for handling large scale datasets as it has the least computational complexity and incurs high costs of implementation, with no mechanism to correct the misclassification of data, if any. Also, the clusters in this case tend to form spherical shapes and show reversal phenomenon, this tends to distort the hierarchical structure.

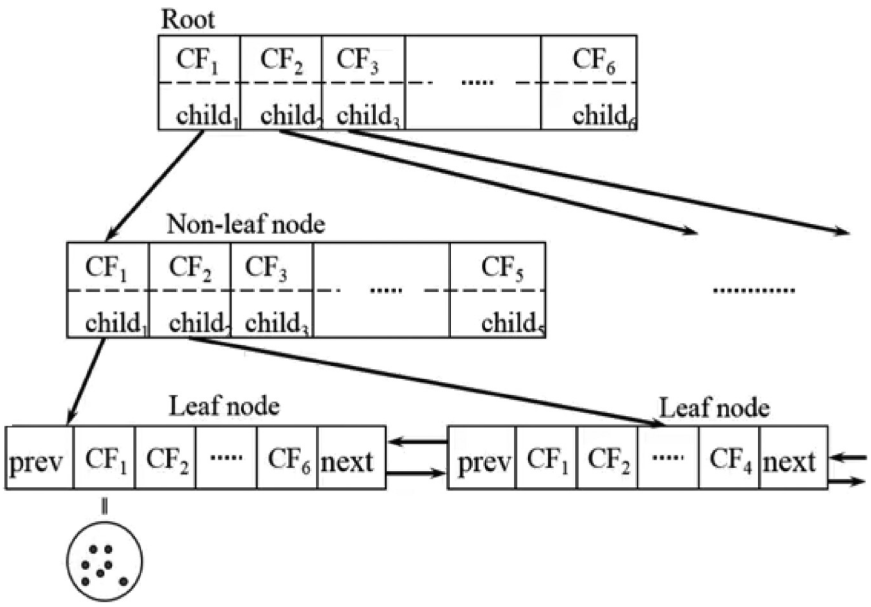

Root
CF1 CF2 CF3 ..... CF6
child1 child2 child3 child6
Non-leaf node
CF1 CF2 CF3 ..... CF5
child1 child2 child3 child5
....
Leaf node
prev CF1 CF2 ..... CF6 next
Leaf node
prev CF1 CF2 ..... CF4 next

Fig. 9. BIRCH: A scalable and flexible clustering method.

3.2.2 *Hierarchical Clustering Algorithms to Handle Large Datasets.* We discuss the three major algorithms based on hierarchical approach that can be used to cluster large datasets:

(1) BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies) [187]: It is a multiple clustering algorithm that uses hierarchical clustering approach. It constructs a CF tree incrementally. This is a hierarchical data structure for multiphase clustering. It mainly has two phases. In the first phase, an initial in-memory CF tree is built by scanning the database, where data is compressed in multilevel way, trying to pressure the inherent clustering structure of the data. In the next phase, CF tree's leaf nodes are clustered by employing an arbitrary clustering algorithm. The key idea of this approach is multilevel clustering, which includes low-level microclustering (reduce complexity and increase scalability) and high-level macroclustering (leave enough flexibility for high-level clustering). This approach scales linearly, which means a single scan helps in finding good clustering, and little more additional scans help in improving the quality. Figure 9 shows how CF trees are used to form clusters using BIRCH. There are certain disadvantages of using BIRCH as well. These are mainly concerns associated with the BIRCH tree. It is sensitive to insertion order of data points and due to leaf nodes fixed size, the natural clusters may not be formed. The clusters formed by using this approach tend to be spiral, given that the major parameters are radius and diameter measures.
(2) CURE (Clustering Using Well Scattered Representatives) [76]: It is used to represent a cluster, using a set of well distributed representative points. The minimum distance amidst the representative points chosen is the cluster distance. It means that this technique incorporates both the single and average link methodologies. CURE is used to capture clusters of arbitrary shapes by choosing a scatter plot. This is shown in Figure 10. The red points are the representative points in the figure. The points in the data space are

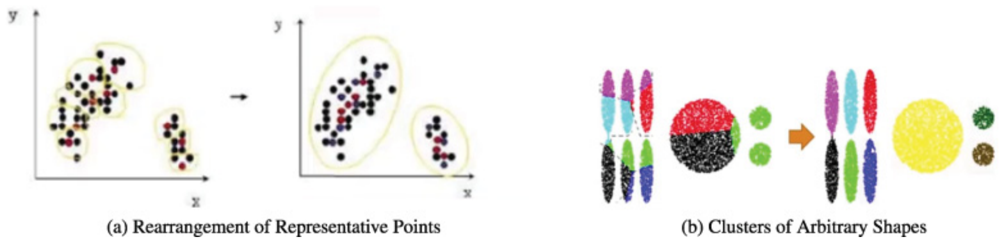
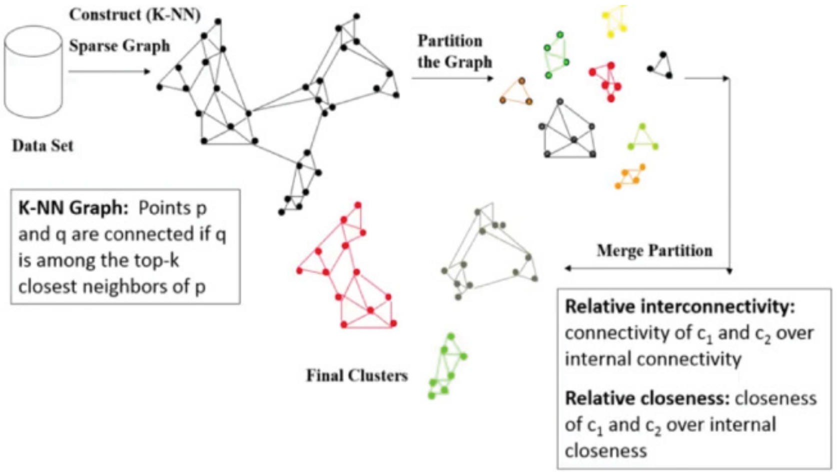

y
x
(a) Rearrangement of Representative Points
(b) Clusters of Arbitrary Shapes

Fig. 10. CURE: Representation of clusters using scattered representative points.



K-NN Graph: Points p and q are connected if q is among the top-k closest neighbors of p

Relative interconnectivity: connectivity of c_1 and c_2 over internal connectivity

Relative closeness: closeness of c_1 and c_2 over internal closeness

Fig. 11. Overall framework of CHAMELEON.

clustered around the centroid, as shown in the first part of the figure. The points shrink while approaching the centroid by a given factor α. It can also be observed that this algorithm has greater effect on outliers, as compared the normal points, i.e., it is more robust to outliers. The outliers lie very far away from the representative points. This not only improves the efficiency, but also the effectiveness of the algorithm. The second part of the figure shows how this algorithm deals with clusters of arbitrary shapes. CURE helps to produce very nicely shaped clusters.

(3) CHAMELEON (Graph Partitioning on the KNN graph of the Data) [97]: It uses a dynamic model to measure similarity between data points. If the closeness (proximity) and interconnectivity amidst any two clusters are high, with respect to the cluster's internal interconnectivity and items' closeness within clusters, then the two clusters are merged into one. This is a two phased algorithm that is graph based. The cluster objects are partitioned into a large count of comparatively small sub-clusters (graphlets) by using graph partitioning algorithm in the first phase. Then, genuine clusters are found by repeatedly combining these sub clusters, by using agglomerative hierarchical clustering algorithm in the second phase. Figure 11 shows the algorithm's overall framework. We use a K-NN graph to represent a large dataset. Then this graph is partitioned into a good count of sub-clusters. After merging these partitions these clusters are merged into high quality clusters. The clusters are merged using relative connectivity and relative closeness measures. CHAMELEON is

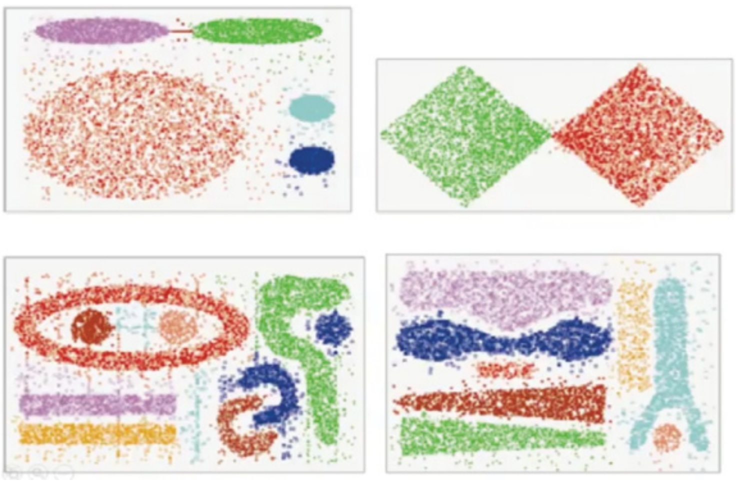


Fig. 12. Quality of clusters produced by CHAMELEON.

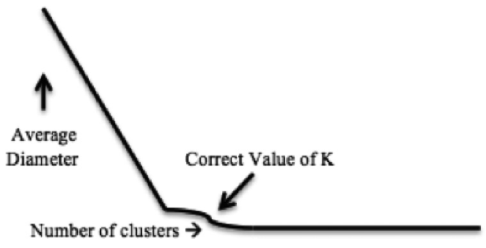


Fig. 13. Average diameter rises when the number of clusters falls below the real number [15].

used to cluster complex objects as shown in Figure 12. It produces quality clusters having different densities, shapes and separates noise points. The figure shows the quality of the clusters generated by using CHAMELEON. It keeps the spirit of hierarchical clustering but uses graph based methods to improve the quality of the clusters produced.

3.2.3 K-means Clustering Approach. *K*-means algorithm is the point assignment type of approach that makes an assumption about the initial cluster count. Then iterative fusions or divisions give the optimal number of clusters. If criteria such as the average diameter or radius are taken for calculating the distance measure between the two clusters, then the value grows till the assumed numbers of clusters remain equal or more than the actual count of clusters obtained. If assumed clusters are less than the real number of clusters, the value rises dangerously as shown in Figure 13. For selecting the right value k, values such as 1, 2, 4, 8 can be taken, till the two values v and $2v$ are obtained. The decrease in the average radius or diameter among these values is very small. The value of k lies between $v/2$ and v. Now, binary search can be used to locate the value of k. Taking total clusterings as $2log_2v$, the best match of k can be found in (log_2v) clustering operations. After the initial value of k is selected, these points are made centroids of the clusters. The remaining points are then represented as "p." Then, these points are checked using a for loop, which is the heart of this approach. For each p, the centroid closest to it is found and p is added to the cluster of the respective centroid. The centroids are also adjusted to account for p [158]. This brute force method is quite inefficient as the cost of computation becomes manyfold when this approach is implemented. Even in a small-scale clustering problems the number of partitions become exceedingly large [115]. This approach works best with compact and hyper spherical cluster shapes. Since it has an optimal time complexity of $O(NKd)$, The massive datasets may be clustered using K-means but after introducing heuristics into this approach.

Algorithm of Bradley, Fayyad, and Reina (BFR) [109] is an intelligent variant of the K-means algorithm, used to cluster huge amounts of data. An assumption regarding shape of clusters distributed normally about a centroid is made. For different dimensions, the cluster's mean and standard deviation may vary. This approach begins by selecting k points. All the points are then read in chunks. These chunks are then partitioned into appropriate sizes. The sizes are selected so that they can be processed in the main memory. The main memory also contains summaries of the k clusters along with other data. This data consists of three object types:

(1) The discard set: These can be seen as meta-data of clusters having simple summaries. The points that are represented by the summary are disposed off and they are not represented in the main memory.
(2) The compressed set: These sets have summaries of the points in approximation with each other. These points are not close to any cluster. Also, the points are not explicitly present in main memory.
(3) The retained set: This set contains points, that can neither be represented by a compressed set nor assigned to a cluster. These points are stored in the main memory in the same form as they are in the input file.

In BFR, data is processed as discussed below:

(1) All the points that are near the centroid of a cluster are added to it.
(2) The points that are not close to any centroid, are clustered in the set that is retained. For this purpose, any clustering algorithm like hierarchical clustering can be employed. The summaries of clusters are added to the compressed set. Single point clusters are allotted to retained set.
(3) Now, new points and the previous retained sets forming mini-clusters are grouped. Also, mini-clusters are formed from old compressed sets. These cannot merge with any of the k-clusters but can merge with one another.
(4) Points that are not in the retained set, i.e., assigned to a mini-cluster or a cluster are assigned to the secondary memory.
(5) For the last chunk of input data, compressed and retained sets need to be handled. They can be treated as outliers and never clustered again, or each retained set's point can be assigned to the cluster of the nearest centroid. Each mini-cluster can now be combined to the cluster having its centroid nearest to the mini-cluster's centroid.

The decision regarding the distance of a point p from a given cluster can be calculated using any of the techniques used to control the value of K in K-means algorithm.

Dimensionality reduction techniques such as PCA can be used with K-means clustering to perform unsupervised clustering tasks [53]. The authors have derived new lower bounds for k-means clustering. It is also found that PCA, via SVD provides the best low-dimensional linear approximations of data. They have analyzed DNA gene expression and internet news groups in order to verify their hypothesis.

3.2.4 Random Sampling Approach. In order to deal with dynamic data applications, we have to opt for randomized sampling. Here, appropriate samples are chosen so that important geometrical properties of the data clusters are maintained. Also, lower limit bound of the minimum sample size is given so as to lower the probability that important points are missed in the cluster sample [76]. The examples of such an approach are CURE and CLARA. CURE is used to cluster large datasets. The shape of the clusters require no initial assumptions. The data distribution can be in the form of rings, S-shapes, or strange bends. The clusters are represented by a group of representation points instead of centroids [85]. The first step of the CURE algorithm is to make clusters from small data sample size in the main memory. Though any clustering approach can be used, hierarchical clustering is mostly applied in theory. After the initial clusters have been formed, representative points are formed by selecting certain points from each cluster. These points should have maximum possible distance between them. The representative points are then moved to a fixed distance between their current position and the centroid of their cluster. In the completion phase of the CURE algorithm, any two clusters will merge with each other if their representative points are sufficiently close to each other. The definition of "close" is a distance provided by the user. The merger continues until no sufficiently close clusters are left. P point assignment is the last step of CURE. Every point p is extracted from the secondary memory to compare with the representative points. Finally, the cluster of the representative point closest to point p, is assigned to point p [76]. CLARA is also a variation of this type of scheme, but the difference is that CLARA forms clusters around a medoid [98].

3.2.5 Randomized Search Approach. This is a very interesting approach in which one can visualize the search being done as equivalent to traversing a graph. The search begins randomly by selecting an arbitrary data item and designating it as the current node. Then, the nearest neighbors of this node are searched in order to find the one with the lowest cost. Now, this becomes the current node and the entire process is repeated. Each node here corresponds to k medoids [133]. CLARANS is the example of such an approach [134]. The time complexity is $O(n)$. This is a better approach than CLARA but its computational time is still quadratic.

3.2.6 Condensation Based Approach. This approach gives a condense summary of the original data. The best example of this type of large scale clustering is BIRCH. It not only has the effective power to deal with massive datasets, but can also efficiently deal with outliers. This approach introduces a new data structure known as the CF tree. The vertices of this structure are defined as $[CF_i, child_i]$, where CF_i represents the cluster i and is defined as $CF_i = (N_i, LS, SS)$ where N_i represents the count of data objects in the cluster, SS is squared summation of objects, and LS is the sum of objects in a linear manner. The leaves of the structure also follow restriction represented as [5]

$$\sum_{i=1}^{Ni}\sum_{m=1}^{Ni}(x_1 - x_m)^2/N_i(N_i - 1)^{1/2}). \tag{44}$$

Important information is captured effectively while reducing the storage required. This algorithm successfully eliminates outliers. Once the CF tree is built, the enabling global clustering and refining of the given clusters is performed by agglomerative hierarchical clustering.

3.2.7 Density-Based Approach. In this approach, it is assumed that the generation of data objects is done by adopting various probability distributions. Different types of density functions are used in order to derive them. If the distribution is known then it becomes simple to find clusters in the given data. If prior probability for cluster $C_i.i = 1, \ldots\ldots, k$ and conditional probability density $P(x|C_i, \theta_i)$ is given θ_i being the unknown parameter vector, whole data's

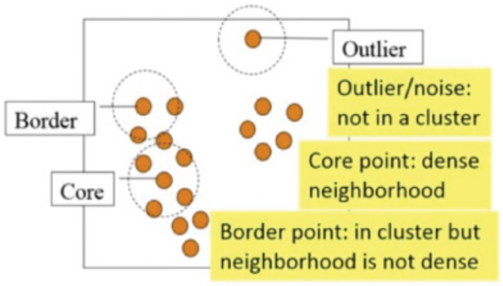

Outlier
Outlier/noise: not in a cluster
Border
Core point: dense neighborhood
Core
Border point: in cluster but neighborhood is not dense

Fig. 14. DBSCAN: The Algorithm.

mixture probability density can be represented as [65, 191]:

$$P(x|\theta) = \sum_{i=1}^{k} p(x|C_i, \theta_i)P(C_i). \tag{45}$$

$$\text{Here, } (\theta = (\theta_1, \ldots\ldots, \theta_k) \text{ and } \sum_{i=1}^{k} P(c_i) = 1.$$

Arbitrary shaped clusters are discovered by this approach to handle noise effectively. It just performs one scan to examine the local region in order to justify density and need density parameters as termination conditions. Examples for this type of clustering approach are DBSCAN [64] and DENCLUE [86]. DBSCAN (Density Based Spatial Clustering Algorithm) states that if an object is a member of a cluster, then its neighborhood density should be reasonably high. A data object in this approach creates new data objects by absorbing all objects in its neighborhood that are created by fulfilling some user defined density threshold. The data structure used by this approach is R*- tree structure. Arbitrary shaped clusters are discovered by this approach and a maximal set of density connected points are defined as a cluster. A point p is selected arbitrary and all points are density reachable from p with respect to Eps (Maximal radius of neighborhood) and MinPts (Minimal count of points in the point's EPS neighborhood) are retrieved by it. A cluster is composed of p as a core point, no points are density reachable from p if it is a border point, and the next point of the database is visited by DBSCAN. The process continues until the processing of all the points have been done. Figure 14 shows how DBSCAN algorithm works. A recent research has pointed out that the claim made by the original paper about the time complexity being $O(n \log n)$ is actually a disclaim [71]. The authors claim that the complexity of the algorithm is $O(n^2)$. It has been further proved that for dimensionality greater than 3, DBSCAN requires $\Omega(n^{\frac{4}{3}})$. A new notion of p-approximate DBSCAN is proposed that is believed to replace DBSCAN on big data due to later's computational intractability. It is also shown that running time can be brought down to $O(n)$, regardless of the dimensionality d. DENCLUE, on the other hand, generates clusters that are created by taking the maximum value of the density function. It greatly influences the data objects in the neighborhood in the data space corresponding to it.

3.2.8 Grid-based Approach. Grid-based clustering explores multi resolution grid data structure in clustering. The data space is partitioned into a definite number of cells forming a grid structure. Then, it finds clusters (dense regions) from the cells in the grid structure. In this approach, number of cells should be less than number of data points for obtaining high efficiency and scalability. It is limited by predefined cell sizes, borders, and density threshold and is not the right approach to handle high density data. It can be explained by algorithms such as WaveCluster [159] and Fractal

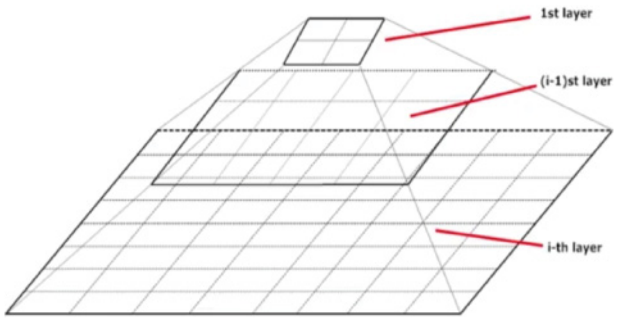


Fig. 15. Spatial area for STING approach [177].

Clustering (FC) [22]. Wave-Cluster defines a set of units in the given feature space and assigns data objects to these units. It further utilizes these units' wavelet transforms. This helps in mapping objects to a frequency domain. The objects are easily distinguishable in the transformed space. FC represents clusters as cells in a grid. The data objects are added into these clusters in an increment manner. Initially, a process is specified that facilitates this addition. There is a predefined condition of relative stability of cluster's fractal dimension. Other approaches that are based on this type of approach are STING [177] and CLIQUE [11]. The spatial area is divided into rectangular cells by a Statistical Information Grid Approach (STING) at different levels of resolution, forming a tree structure. Figure 15 shows the spatial area's division into rectangular cells for STING analysis. This means that a cell at a higher level contains various next lower level smaller cells. Each cell's statistical information is being computed and stored in advance for answering the queries. Higher cell parameters can be easily computed from the parameters of the lower cells. The region query process starts at the root and proceeds to the next lower level using the STING index. Then it calculates the likelihood of a cell's relevance to query at some confidence level using the cell's statistical information. Only children of likely relevant cells are recursively explored. The process is iterated till the attainment of bottom layer. This is a query independent, easy to parallelize, incremental approach but its probabilistic nature may imply a loss of accuracy in query processing. CLIQUE (Clustering In QUEst) is a clustering algorithm based on density as well as grid subspace. The data space is discretized through grid where, a grid estimates the density by the count of points present in a grid cell. The cluster formed is a maximum set of connected dense units in a subspace. If the total data point fractions included in unit outstrips the input model parameter then the unit is dense. A neighborhood dense cells set in an arbitrary subspace is referred as a subspace cluster having clusters' minimal descriptions. The CLIQUE algorithm identifies the subspaces that contains clusters. Then, it identifies the clusters and generates minimal description of the clusters. This is a very efficient approach but resultant quality critically is dependent on the appropriate selection of the number as well as width of the partitions and grid cells. Figure 16 shows application of CLIQUE with respect to three variables: salary, age, and vacation. In one-dimensional spaces the numerical intervals are discretized and each of the axis of the grid are considered. Dense regions (clusters) in each subspace are found and there minimal description is generated. Then all these three variables are demonstrated in a 3-D space.

3.2.9 Probabilistic Model-based Clustering Methods. A probabilistic model gets data from a generative process. It assumes that data is produced by a combination of underlying probabilistic distributions. It attempts to optimize the fit amidst observed data and any mathematical model using probabilistic methods. In probable model based clustering, the mathematical representation

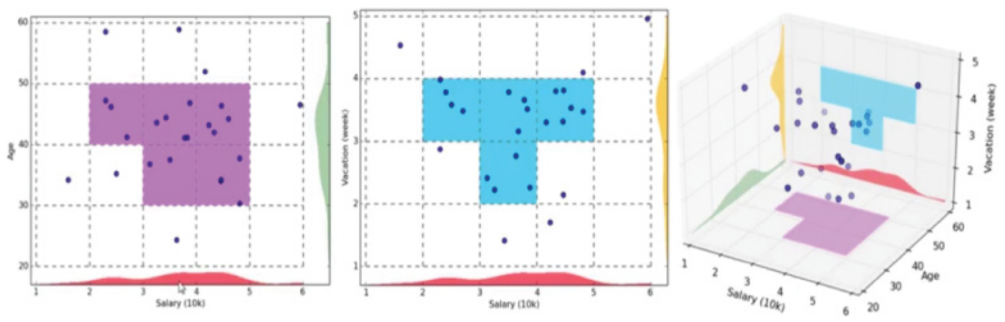


Fig. 16. Application of CLIQUE.

of each cluster can be done using parametric probability distribution (e.g., Gaussian or Poisson distribution). The clusters will be formulated of the data points that most likely belong to the same distribution. There also has to be some parameter estimation, so that the data points will have a maximal tendency for fitting in the model by a mixture of K component distributions (i.e., K clusters) [51]. Some of the probabilistic models are discussed below:

(1) Mixture model for cluster analysis [122]: A set C with K probabilistic clusters $C_1, \ldots, C_k$ having $f_1, \ldots, f_k$ as probability density functions respectively and $\omega_1, \ldots, \omega_k$ as their probabilities. Then, an object's probability can be generated by cluster C_i as [122]

$$P(O, C_j) = P(C_j)P(O \mid C_j) = \omega_j f_j(O). \tag{46}$$

Probability of O, that is produced by cluster set C is

$$P(O \mid C) = \sum_{j=1}^{k} \omega_i f_j(O). \tag{47}$$

It is assumed that objects are generated in an independent manner for a dataset $D = O_1, \ldots, O_n$:

$$P(D \mid C) = \prod_{i=1}^{n} P(O_i \mid C) = \prod_{i=1}^{n} \sum_{j=1}^{k} \omega_i f_i(O_i). \tag{48}$$

The task is to find K probabilistic cluster set named as C so that $P(D \mid C)$ is maximized. It is mostly intractable to maximize $P(D \mid C)$ because an arbitrarily complicated form can be taken by a cluster's probability density function. It is assumed that the probability density functions are parameterized distributions for making it computationally feasible (as a compromise). A set of K probabilistic clusters are inferred, most likely for generating D. The values of the discrete latent variables can be interpreted as the assignments of data points to specific components (i.e., clusters) of the mixture. A parametric distribution is used to mathematically represent each cluster. In principle, any component type can be used to construct the mixtures and still perfectly good mixture models can be obtained. In practice, a lot of effort is made over to parametric mixture models where, all components are from the same parametric distribution family, having dissimilar parameters. For example, all Gaussian mixtures with different means and variances, all poisson distributions with different means or all power laws with different exponents. Two most common mixtures are mixture of Gaussian (continuous) and mixture of Bernoulli (discrete) distributions.

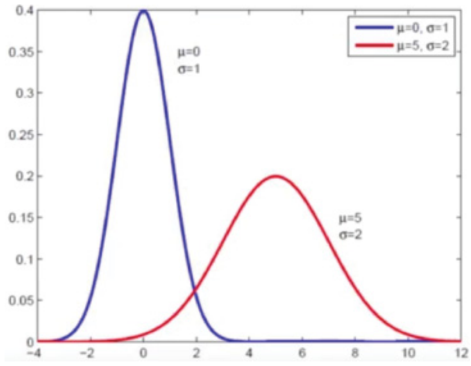


Fig. 17. Plots of univariate Gaussian distribution for various parameters of μ and σ.

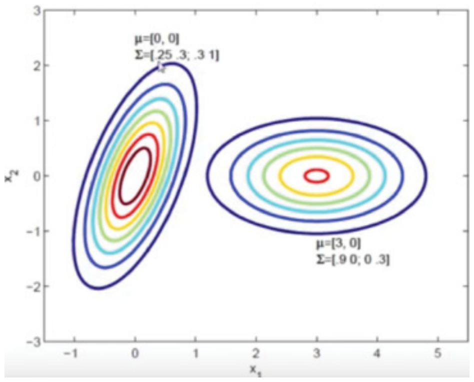


Fig. 18. Contours of multivariate Gaussian distribution for various parameters of μ and Σ.

(2) Gaussian mixture models [34]: It is assumed that all the clusters are formed with Gaussian distribution with different parameters in this model. Gaussian distribution is assumed for both single and multivariate distributions. Figures 17 and 18 show various plots and contours for Gaussian distributions for various parameters. Every cluster C_i is assumed to be characterized using multivariate normal distribution given as

$$f_1(x) = f(x \mid \mu_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^d \mid \Sigma_i \mid}} exp\left(-\frac{(x-\mu_i)^r \Sigma_i^{-1}(x-\mu_i)}{2}\right), \tag{49}$$

where cluster mean is μ_i and covariance matrix is Σ_i. These are unfamiliar parameters and $f_i(x)$ is the cluster C_i probability density. It is assumed that the probability density function of X can be represented as Gaussian mixture model (GMM) over all the K cluster normals defined as

$$f(x) = \sum_{i=1}^{k} f_i(x)P(C_i) = \sum_{i=1}^{k} f(x \mid \mu_i, \Sigma_i)P(C_i), \tag{50}$$

where the prior probabilities known as mixture parameters $P(C_i)$ must assure

$$\sum_{i=1}^{k} P(C_i) = 1.$$

Maximum Likelihood Estimation (MLE). For any data set D, the model parameters θ's likelihood is

$$P(D \mid \theta) = \prod_{f=1}^{n} f(x_j), \tag{51}$$

or written as

$$In\ P(D \mid \theta) = \sum_{j=1}^{n} Inf(x_j) = \sum_{f=1}^{n} In \sum_{i=1}^{k} f(x_j \mid \mu_i, \Sigma_i)P(C_i). \tag{52}$$

MLE is to choose parameters θ: $\theta^* = \arg\ max_\theta\{P(D \mid \theta)\}$, or maximize the log likelihood, θ: $\theta^* = \arg\ max_\theta\{InP(D \mid \theta)\}$, directly maximizing the log likelihood over ∂ is difficult. EM approach is used for searching parameters ∂'s maximum likelihood estimate.

Expectation Step: Given current estimates for ∂, compute the cluster posterior probability $P(C_i \mid x_j)$ via Bayes theorem represented as

$$P(C_i \mid x_j) = \frac{f_i(x_j) \cdot P(C_i)}{\sum_{a=1}^{k} f_a(x_j) \cdot P(C_a)}. \tag{53}$$

This theorem tries to find out the probability of the cluster C_i under the condition x_j by using probability density function $f_i(x_j)$ with probability of cluster C_i divided by the summation of all probability density functions $f_a(x_j)$ for all values of "a" from 1 to K and probability to generate C_a.
Maximization Step: Weight $P(c_i \mid \mu_j)$ re-estimate ∂ is used, i.e., re-estimate μ_i $\sum_i$ and $P(C_j)$ for each cluster C_i.

(3) The Expectation–Maximization Algorithm: This algorithm is used for a single variable or univariate. We can view the existing algorithms under the EM framework. For example, the K-means algorithm can be viewed under the EM algorithm as [79]:
  (a) Expectation Step (E-Step): The cluster having its center nearest to the object is assigned each object. It is expected that an object correlates to the closest cluster.
  (b) Maximization Step (M-Step): Taking cluster assignment in consideration, the center for each cluster is adjusted by the algorithm so that the summation of distance from the objects assigned to this cluster and the new center is minimized. We try to recalculate the center for each cluster.

  EM Algorithm: A framework for approaching maximum likelihood or maximizing statistical models parameters' *posteriori* estimates.
  (a) E-step: It is used for assigning objects to clusters in accordance with the probabilistic clusters' present parameters. It is a soft assignment.
  (b) M-Step: It searches for new clustering or parameters for minimizing the summation of squared errors (SSE) or expected likelihood.

  The two variants of the EM Algorithm are discussed as follows:
  (a) Expectation maximization algorithm for one dimension (Univariate) [57]: Considering a dataset D inhering a single attribute x, with each point x_i ($i = 1, \ldots, n$) being a random sample from x. The univariate normals for each cluster are used in mixture model, given in normal Gaussian distribution form as

$$f_i(x) = f(x|\mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi\sigma_i}} exp\left\{-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right\}. \tag{54}$$

  Initialization for each cluster parameter C_i with $i = 1, \ldots, k$, randomly is done as μ_i is selected uniformly at random, $\sigma_i^2 = 1$. The probability $P(C_i) = \frac{1}{k}$, this means that each cluster is having the same probability.

  The posterior probability $(P(C_i|x_j))$ in the expectation step can be computed as

$$P(C_i|x_j) = \frac{f(x_j|\mu_i, \sigma_i^2).P(C_i)}{\sum_{a=1}^{k} f(x_j|\mu_a, \sigma_a^2).P(C_a)}. \tag{55}$$

  For maximization step, the maximum likelihood estimates of the cluster parameters is computed. This is done by re-estimating μ_i, σ_i^2 and $P(C_i)$ for each cluster C_i. The execution of EM algorithm for univariate (single dimension) for GMM iteration at 11 is shown in Figure 19.
  (b) Expectation maximization algorithm (Multivariate) [51]: This algorithm has three steps discussed below:

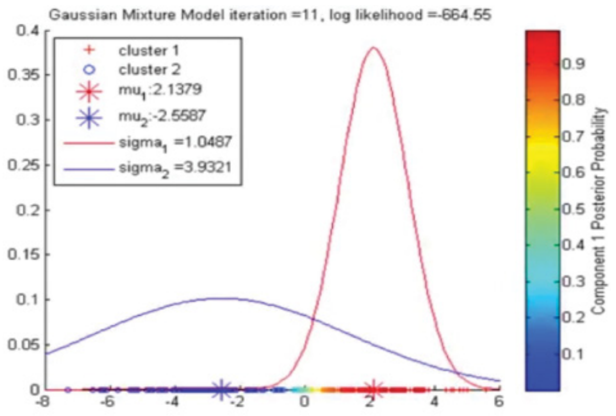


Fig. 19. Execution of EM Algorithm for univariate (single dimension).

- Initialization: Randomly initialize $\mu_1, \ldots, \mu_k$; $\sum_i \leftarrow 1 \forall\, i = 1, \ldots, k$; $P(C_i) \leftarrow \frac{1}{k} \forall\, i = 1, \ldots, k$. This is kept in repeat until the means across two iterations is not greater than threshold ϵ.
- Expectation Step: Objects are assigned to clusters in accordance with the probabilistic clusters' current parameters. That means that for all the clusters $i = 1, \ldots, k$, and for all the objects $j = 1, \ldots, n$, the posterior probability $P(C_i|x_j)$ can be expressed as
$$w_{ij} \leftarrow \frac{f(x_j|\mu_i, \sum_i) \cdot P(C_i)}{\sum_{a=1}^{k} f(x_j|\mu_a, \sum_a) \cdot P(C_a)}. \tag{56}$$
The numerator is the product of the density function and the probability and the denominator is calculated for every cluster and then it is summed up.
- Maximization Step: It aims at finding new clustering or parameters that minimizes SSE or the expected likelihood. For all $i = 1, \ldots, k$ do, until the change is very small.
  - Re-estimate Mean:
$$\mu_i \leftarrow \frac{\sum_{j=1}^{n} w_{ij}.x_j}{\sum_{j=1}^{n} w_{ij}}. \tag{57}$$
  - Re-estimate Covariance Matrix:
$$\sum_{i=i}^{n} \leftarrow \frac{\sum_{j=1}^{n} w_{ij}(x_j - \mu_i)(x_j - \mu_i)^T}{\sum_{j=1}^{n} w_{ij}}. \tag{58}$$
  - Re-estimate priors:
$$P(C_i) \leftarrow \frac{\sum_{j=1}^{n} w_{ij}}{n}. \tag{59}$$

The execution of 2-D dataset is given in Figure 20. It shows the initial dataset and its formation of clusters in the corresponding iterations.

Analysis of Mixture Model Methods [26]: We start by discussing the relationship between the K-means and the EM algorithms. K-means can be considered as a special case of EM algorithm, where
$$P(x_j|C_i) = \begin{cases} 1 & if\ C_i = \arg\min_{c_a}\{||x_j - \mu_a||^2\} \\ 0 & otherwise \end{cases}$$

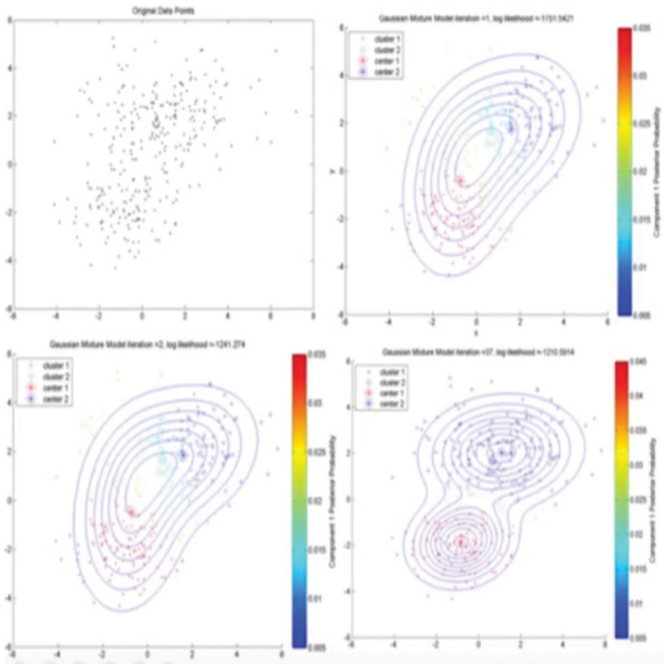


Fig. 20. Execution of EM Algorithm for multivariate (two dimensions).

$$P(C_i|x_j) = \begin{cases} 1 & if\ x_j \in C_i, i.e., C \ = \ \arg\min_{c_a}\{||x_j - \mu_a||^2\} \\ 0 & otherwise \end{cases} \tag{60}$$

K-means is viewed as a hard version of the EM algorithm. In the E step, local minimization is taken instead of a distribution. The GMM is the soft version of *K*-means. The distribution is calculated in the E-step in this case, and weighted sum is used to compute new centers in the M-Step. GMM introduces variance to learning, whereas clusters in *K*-means have the same variance. Compared with the *K*-means algorithm, the EM algorithm for GMM takes many more iterations to reach convergence. To find a suitable initialization, and speed up the convergence for GMM, we need to first run the *K*-means algorithm. Then, the cluster means and covariances are chosen and the data point fractions are assigned to the respective clusters for initializing μ_k, $\sum_k$ and $P(C_i)$, respectively. When a Gaussian component collapses onto a particular data point, it is called singularity. As soon as a Gaussian component collapsing is detected its mean and covariance are reset and the process of optimization continues.

Mixture models are more generic when compared with partitioning and fuzzy clustering. A small number of parameters are used to characterize clusters. The generative models or the statistical assumptions are satisfied by the results. There are a number of weaknesses for this type of model. They converge to local optimal, which can be overcome by running it a multiple number of times with random initializations. The estimation of the count of clusters is very hard as mixture models work on large datasets and it is hard to estimate the number of clusters.

3.2.10 Clustering Graphs and Network Data. Most real world data is interconnected forming gigantic networks or graphs. Networks can be homogenous or heterogenous. Homogeneous networks have vertices and edges of one type of data, e.g., web search engines, co-author graphs, friendship networks, social networks, and so on. Heterogenous networks, on the other hand have vertices and edges of multiple types. Like two typed graphs (e.g., authors and conferences, customers and products) that have two attributes that can be linked together, multi-typed graphs have multiple type nodes and edges (e.g., research networks (authors, papers, publishing houses, etc.); medical networks (doctors, patients, drugs, treatments); freebase (different types of entities interacting with each other)). There is a distinction between clustering, graph partitioning and community discovery. Clustering groups objects into groups that can be hard like K-means or soft like EM; complete or partial; balanced or skewed. Graph partitioning is hard grouping that is complete and typically balanced. It means that the smaller graphs are as complete and balanced in themselves, as a singleton graph. Community discovery can be partial, often interested only in finding the densely connected components and not in the cluster assignment of every vertex. In this section, we discuss the basic concepts of graphs, evaluation measures, and some popular graph based clustering approaches.

— *Graphs, networks and their representations:* A network or a graph is represented as $G = (V,E)$, where a set of vertices or nodes is denoted by V, and a set of edges/links is denoted by E. If there exits more than one edge between the same pair of vertices, it is called multi-edge graph and if a vertex is connected to itself by an edge, then it is called a loop. If a network neither has self-edges or multi-edges, it is called a simple network. There are various ways to represent a network like [155]: Adjacency matrix ($A_{ij} = 1$, if there is an edge between vertices i and j; 0 otherwise); directed graph (if each edge has a direction ($tail \rightarrow head$, i.e., $A_{ij} = 1$, if there is an edge from j to i; 0 otherwise); weighted graph (if a weight w_{ij} usually a real number is associated with each edge v_{ij}). The vertex degree can also be defined for undirected and directed networks. We consider a network $G = (V, E)$. For a undirected network, degree of a vertex $d(v_i)$ is given by

$$d(v_i) \ = | \ v_j \ | \ \ s.t. \ e_{ij} \ \in \ E \ \wedge e_{ij} = e_{ji}. \tag{61}$$

In directed networks, in degree of a vertex $d_{in}(v_i)$ is given by

$$d_{in}(v_i) \ = | \ v_j \ | \ \ s.t. \ e_{ij} \ \in \ E. \tag{62}$$

Out degree of vertex $d_{out}(v_i)$ is given by

$$d_{out}(v_i) \ = | \ v_j \ | \ \ s.t. \ e_{ji} \ \in \ E. \tag{63}$$

— *Typical evaluation measures for graph clustering:* Graph clustering can be evaluated using common measures such as Mincut, Ratio cut, Normalized cut, Conductance, and Modularity. Typical similarity measures across networks are SimRank and Personalized page rank. Now, we introduce the graph cutting method known as minimum cut (mincut). The general philosophy is to cut minimum number of edges to partition the graph into two. Given the number "k" of partitions, choose a partition $C_1, \ldots, C_k$ that is minimum where $W(C_i, \overline{C_i})$ is the sum of weights of the edges connecting C_i and those in other partitions, is given by [67]

$$Cut(C_1, C_2, \ldots, C_K) = \frac{1}{2} \sum_{i=1}^{k} W(C_i, \overline{C_i}). \tag{64}$$

Mincut can be a poor cut in certain cases like if we cut one node from the remaining graph. The motivation is to make partitions balanced. The traditional approach uses Spectral Clustering to solve the graph cut problem. There are other graph cutting measures as well [55]:

(1) Ratio Cut: The sum of the edge weights connecting a cluster C to the remaining graph that is normalized by the size of C. It is given by

$$RatioCut(C_1, C_2, \ldots, C_k) = \frac{1}{2}\sum_{i=1}^{k}\frac{W(C_i, \overline{C_i})}{\mid C_i \mid} \tag{65}$$

(2) Normalized Cut [36]: The sum of the edge weights connecting C to the remaining graph normalized by cluster C's total degree is known as a normalized cut. The lower the normalized cut, the better the community, i.e, among themselves, they are being well connected and connected sparsely to the remaining graph. It is given by

$$NCut(C_I, C_2, \ldots, C_k) = \frac{1}{2}\sum_{i=1}^{k}\frac{W(C_i, \overline{C_i})}{d \mid C_i \mid}. \tag{66}$$

(3) Conductance: It is similar to normalized cut and is given by

$$Conductance(C) = \frac{\sum_{i\in C, j\in \overline{C}} W(i,j)}{min(\sum_{i\in C} d(i), \sum_{j\in \overline{C}} d(j))}. \tag{67}$$

The sum of conductance of a partition of a graph into k clusters $C_1, C_2, \ldots, C_k$ being the sum for normalized cut or conductance of individual partitions C_i for $i = 1, \ldots, k$.

(4) Modularity [164]: It is the sum of differences, for each cluster, between fraction of internal edges and fraction of edges that are expected to be inside a random cluster with the same total degree. This means that the modularity of clustering of a graph is the difference between the fraction of all edges that fall into individual clusters and the fraction that would do so if the graph vertices were randomly connected. It is represented by

$$Q = \sum_{i=1}^{k}\left[\frac{W(C_i, C_i)}{e} - \left(\frac{d(C_i)}{2e}\right)^2\right], \tag{68}$$

where C_is are clusters, e is the number of edges, and $d(C_i)$ represents the total degree of cluster C_i. For K clusters, the modularity of a cluster assesses the quality of the clustering. The optimal clustering of graphs maximizes the modularity. Unfortunately, optimizing any of these normalized objective function is NP-hard.

(5) SimRank: Similarity Based on Random Walk and Structural Context [94]: It is a similarity measure based simple and intuitive graph model. In order to check similarity between X and Y, the random walk model is considered. Walk in a graph (G) is defined amidst X and Y nodes. There is an ordered sequence of vertices from the node X to Y, so that each consecutive vertex pair contains an edge among them. SimRank is dependent on its neighbors similarity, i.e., structural context similarity. In a directed graph (G=(V,E)), neighborhood is either in-neighborhood of V$(I(V) = \{U|(U,V) \in E\})$ or out-neighborhood of V $(O(V) = \{W|(V,W) \in E)$. Alike, defined by SimRank, neighborhood (where C is a constant between 0 and 1) is given by

$$s(u,v) = \frac{C}{\mid I(u) \mid\mid I(v) \mid}\sum_{x\in I(u)}\sum_{y\in(v)} s(x,y), \tag{69}$$

where S represents similarity. Initialization is done as

$$S_i(u,v)\begin{cases}0 & if \quad u \neq v\\ 1 & if \quad u = v\end{cases} \tag{70}$$

After this S_{i+1} is computed from S_i based on the definition. It is costly to compute SimRank but many efficient computation methods have been proposed.

— *Approaches for graph clustering in homogenous networks:* We discuss basic concepts and approaches dealing with clustering formulated in networks formed by homogeneous data.

(1) Partition with geometric information: It is applied on graphs whose geometric information (i.e., meshes) is known. The aim of this type of clustering is to have points geometrically close to each other in one cluster and those far apart in different clusters. In many cases, this clustering is defined by space decomposition. An example of this type of clustering is FADE [149]. This algorithm is used for 2-D drawing, geometric clustering, and viewing large undirected graphs in a multilevel space. It uses a decomposition tree for its implementation.

(2) Graph growing and greedy algorithms: Clustering with good quality scores can be obtained based on greedy agglomeration [59]. In this case, the modularity approach is used most of the times. The process starts with single clusters and iteratively merges those two clusters that have best modularity. This means that largest increase or smallest decrease is chosen. After n–1 merges have taken place, the highest modularity is returned. The graph growing and greedy algorithm maintains a symmetric matrix. The entries of the matrix are represented as

$$m_{i,j} = q(C_{i,j}) - q(C). \tag{71}$$

Here, C is the current clustering, $m_{i,j}$ is a symmetric matrix, $C_{i,j}$ is obtained from C after merging C_i and C_j. There are many pairs i and j such that $m_{i,j}$ is maximum. The algorithm selects random pairs. The upper bound on the number of iterations is given by n–1. The algorithm is terminated as soon as the matrix m contains only non-positive entries. It should be noted that it is NP hard to maximize modularity in general graphs, hence the greedy algorithms are not optimal. [73] have proposed the randomized greedy modularity clustering algorithm and the core group graph clustering schemes. This approach has been able to able to achieve highest modularity, and tradeoff between modularity and performance. It finds locally optimal solutions. Further, the similarity between randomized greedy models and incomplete solvers is found. All of this is achieved using graph theory as the fundamental tool of data visualization.

(3) Agglomerative and divisive clustering: This type of clustering is based on the principle of merging small communities if it increases the graph modularity. Agglomerative approach starts with every vertex in a separate cluster and keeps merging clusters until the clustering can no longer be improved. Conversely, the divisive approach removes edges to detect the communities. They start with the entire graph and split clusters until further splitting is no longer possible. This method is implemented by making use of graph partitioning algorithms and techniques [131]. The authors in [38] have proposed a new graph clustering algorithm aimed at obtaining clustering of high modularity. The authors are able to achieve high modularity score, using a divisive clustering approach. The work in [56] shows that the problem of finding a partition maximizing the modularity of a given graph, G, can be

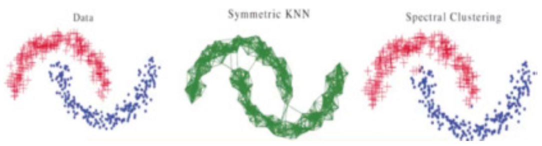


Fig. 21. Steps of spectral clustering.

reduced to a minimum weighted cut problem on a complete graph, having same vertices as G.

(4) Markov's clustering: This type of clustering is based on the concept that "random walk that visits a dense cluster will likely not leave the cluster until many of its vertices have been visited." Markov's clustering modifies the matrix of transition probabilities instead of simulating random walks. If $M(G)$ corresponds to random walks of length at most one, then two operations are applied iteratively [32]:
—Expansion: M is taken to power $e \epsilon N > 1$ in order to simulate e steps of random walk with the current transition matrix.
—Inflation: In this step, M is re-normalized after taking every entry to its rth power, $r \epsilon R^+$

Markov's clustering applies expand and inflate iteratively on the transition probability matrix, it prunes away the smaller values in each column, and re-normalizes it for next iteration.

(5) Spectral clustering: This is one of the most popular clustering methods recently. This type of clustering makes no assumption about the shapes of clusters and can handle intertwined spirals, and so on. Process of spectral clustering starts from construction of similarity graphs (e.g., KNN graph) for all data points. These are based on their neighborhoods, using distances such as Euclidean distance. Then, data points are embedded in low-dimensional space where clusters are more obvious. This can be done with the help of graph Laplacian's eigenvectors. A classical clustering algorithm (e.g., K-means) is applied to partition the embedding [190]. Figure 21 shows the steps of spectral clustering. An adjacency matrix is used to represent a graph. It is a n × n symmetric matrix and is represented as

$$A_{ij} \begin{cases} W_{ij} : weight\ of\ edge\ (i,j) \\ 0 : if\ no\ edge\ between\ i,j \end{cases} \tag{72}$$

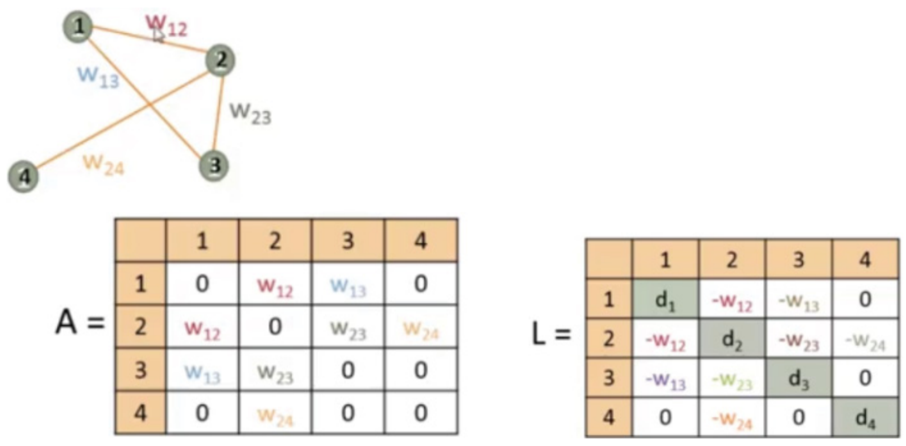
Then, Laplacian is performed on this matrix given by L = D − A, where D being diagonal matrix, and is given as

$$d_i = \sum_{(j|(i,j)\in E)} W_{ij}. \tag{73}$$

L is represented as

$$L_{ij} \begin{cases} d_i : if\ i = j \\ -w_{ij} : if\ (i,j)\ is\ an\ edge \\ 0 : if\ no\ edge\ between\ i,j \end{cases} \tag{74}$$

Figure 22 represents matrix representation of a graph and its Laplace transformation. The values of d are calculated as: $d_1 = w_{12} + w_{13}$
$d_2 = w_{12} + w_{23} + w_{24}$
$d_3 = w_{13} + w_{23}$
$d_4 = w_{24}$



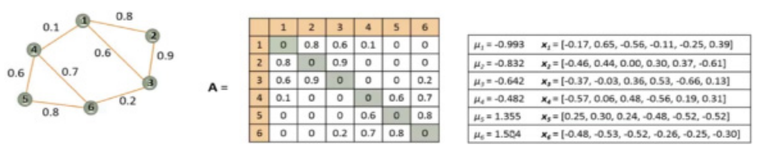
	1	2	3	4
1	0	W_{12}	W_{13}	0
2	W_{12}	0	W_{23}	W_{24}
3	W_{13}	W_{23}	0	0
4	0	W_{24}	0	0

(a) Matrix representation of the graph

	1	2	3	4
1	d_1	$-W_{12}$	$-W_{13}$	0
2	$-W_{12}$	d_2	$-W_{23}$	$-W_{24}$
3	$-W_{13}$	$-W_{23}$	d_3	0
4	0	$-W_{24}$	0	d_4

(b) Laplace Transformation

Fig. 22. Matrix representation of a graph and Laplace transformation.



	1	2	3	4	5	6
1	0	0.8	0.6	0.1	0	0
2	0.8	0	0.9	0	0	0
3	0.6	0.9	0	0	0	0.2
4	0.1	0	0	0	0.6	0.7
5	0	0	0	0.6	0	0.8
6	0	0	0.2	0.7	0.8	0

$\mu_1 = -0.993$	$\boldsymbol{x}_1 = [-0.17, 0.65, -0.56, -0.11, -0.25, 0.39]$
$\mu_2 = -0.832$	$\boldsymbol{x}_2 = [-0.46, 0.44, 0.00, 0.30, 0.37, -0.61]$
$\mu_3 = -0.642$	$\boldsymbol{x}_3 = [-0.37, -0.03, 0.36, 0.53, -0.66, 0.13]$
$\mu_4 = -0.482$	$\boldsymbol{x}_4 = [-0.57, 0.06, 0.48, -0.56, 0.19, 0.31]$
$\mu_5 = 1.355$	$\boldsymbol{x}_5 = [0.25, 0.30, 0.24, -0.48, -0.52, -0.52]$
$\mu_6 = 1.504$	$\boldsymbol{x}_6 = [-0.48, -0.53, -0.52, -0.26, -0.25, -0.30]$

Fig. 23. Graph G with adjacency matrix A, eigenvalues and eigenvectors of A.

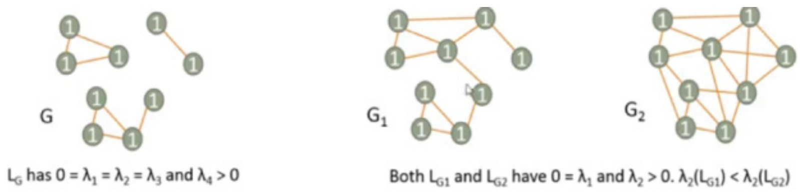
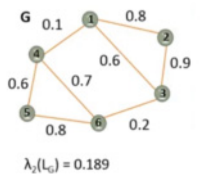

G
G1
G2
L_G has $0 = \lambda_1 = \lambda_2 = \lambda_3$ and $\lambda_4 > 0$
Both L_{G1} and L_{G2} have $0 = \lambda_1$ and $\lambda_2 > 0$. $\lambda_2(L_{G1}) < \lambda_2(L_{G2})$

Fig. 24. Graph Laplacian of G.

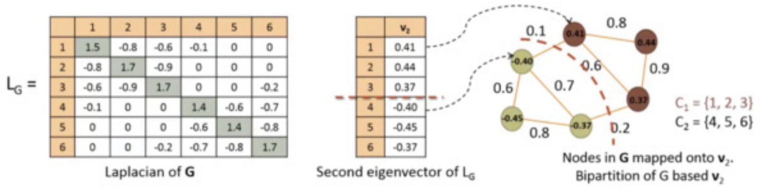
Eigenvalue and eigenvector of graphs are represented as in Figure 23. For a matrix A, λ is an eigenvalue of A if, for some vector v, $Av = \lambda v$. v is the eigenvector of A corresponding to λ. For a graph G, with n nodes, its adjacency matrix has n eigenvalues sorted in descending order. $\{\mu_1, \mu_2, \ldots, \mu_n\}$ are the n eigenvalues, where $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_n$ and n corresponding eigenvectors are $(x_1, x_2, \ldots, x_n)$ [179].

The graph Laplacian of G, L_G also has eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, where $0 = \lambda \leq \lambda 2 \leq \cdots \leq \lambda_n$ and eigenvectors $\{v_1, v_2, \ldots, v_n\}$. Eigenvalues reveal global graph properties that are not apparent from the edge structure. If 0 is the eigenvector of L with K different eigenvectors, i.e., $0 = \lambda_1 = \lambda_2, \ldots, = \lambda_k$, then G has K connected components. If the graph is connected, $\lambda \geq 0$ and λ_2 is the algebraic connectivity of G. The greater λ_2, the more connected G is [99]. The graph Laplacian of G is represented in Figure 24.

To find the bi-partition graph using spectral clustering, we take the second eigenvector of the Laplacian, v_2, corresponding to λ_2, the algebraic connectivity of G. The smaller the λ_2, the better the quality of partitioning. For each node i in



(a) Weighted Graph G


L_G =

	1	2	3	4	5	6
1	1.5	-0.8	-0.6	-0.1	0	0
2	-0.8	1.7	-0.9	0	0	0
3	-0.6	-0.9	1.7	0	0	-0.2
4	-0.1	0	0	1.4	-0.6	-0.7
5	0	0	0	-0.6	1.4	-0.8
6	0	0	-0.2	-0.7	-0.8	1.7

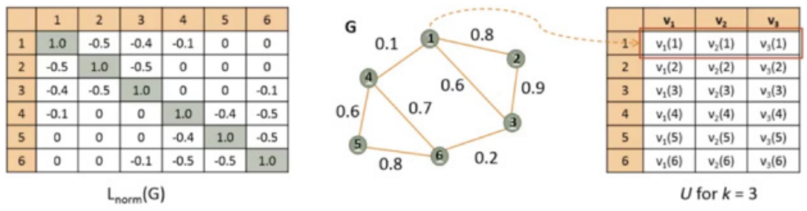
Laplacian of **G**

	v_2
1	0.41
2	0.44
3	0.37
4	-0.40
5	-0.45
6	-0.37

Second eigenvector of L_G

(b) Laplacian of G, second eigenvector of L_G and mapping of nodes in G onto vertices

Fig. 25. Bipartitioning via spectral methods.



	1	2	3	4	5	6
1	1.0	-0.5	-0.4	-0.1	0	0
2	-0.5	1.0	-0.5	0	0	0
3	-0.4	-0.5	1.0	0	0	-0.1
4	-0.1	0	0	1.0	-0.4	-0.5
5	0	0	0	-0.4	1.0	-0.5
6	0	0	-0.1	-0.5	-0.5	1.0

$L_{norm}(G)$

	v_1	v_2	v_3
1	$v_1(1)$	$v_2(1)$	$v_3(1)$
2	$v_1(2)$	$v_2(2)$	$v_3(2)$
3	$v_1(3)$	$v_2(3)$	$v_3(3)$
4	$v_1(4)$	$v_2(4)$	$v_3(4)$
5	$v_1(5)$	$v_2(5)$	$v_3(5)$
6	$v_1(6)$	$v_2(6)$	$v_3(6)$

U for *k* = 3

Fig. 26. Extension of bipartition to *k* partitions.

G assign it the value v_2(i). To find clusters C_1 and C_2 assign nodes with $v_2(i) > 0$ and $v_2(i) < 0$ to C_2 [52]. Bi-partitioning via spectral methods is represented in Figure 25.

Extension of the approach to K partitions is given in Ng-Jordon-Weiss (NJW) algorithm [112]. In this approach the first K eigenvectors $v_1, v_2, \ldots, v_k$ of L_{norm} is calculated. It is called the normalized Laplacian (L_{norm}) and is calculated by: $L_{norm} = D^{-1/2}LD^{-1/2}$. Let $U \in R^{nXk}$ be the matrix containing the vectors $v_1, v_2, \ldots, v_k$ as columns. For $i = 1, \ldots, n$, take the ith row of U as its feature vector after normalizing to norm 1. Cluster the points with k-means into k clusters $C_1, \ldots, C_k$. This is commonly used as a dimensionality reduction technique for clustering. Figure 26 represents this approach using sample values.

Since this algorithm has been used a lot by researchers recently, there have been many interesting new developments in this area. The authors in [135] have used the fact that MinMax cut provides more balanced clusters as compared to other methods. They have applied an additional nonnegative constraint to the MinMax graph clustering to optimize the existing approach. In this method, there is a direct assignment of data clusters to data points. The authors have proved that their method either converges or outperforms the traditional spectral relaxation approach based on ratio and normalized cuts. The authors in [139] have pointed out that the assumption for spectral clustering that two nearby data points in the high density region of a low-dimensional data space lie in the same cluster, does not hold good for high-dimensional clusters. They have proposed a framework for in-sample and out-of-sample spectral clustering. In this framework, a Laplacian matrix generated from local regression of each pattern is incorporated. There have also been some contributions in refinement of Laplacian theory in order to propose more sophisticated algorithms. The authors in [137] have proposed the constrained Laplacian rank algorithm for graph clustering. In this approach, the data graphs are allowed to adjust themselves as part of the clustering process. The constrained Laplacian Rank method learns a graph with exactly K-connected components (where k is the number of clusters). Two versions of this method have been developed based on L-1 and L-2 norms. Hence, this algorithm yields two new graph based clustering objectives. A big challenge of machine learning is to formulate self training clustering method. An approach addressing this challenge has been proposed by the authors in [138]. The data samples are selected with an aim to use them as a training sets to minimize an estimated the Bayes error. Then, semi-supervised learning is explored to perform clustering. A specific regularization framework is developed in order to perform semi-supervised learning. The proposed clustering algorithm can be applied in a semi-supervised setting with partial class labels. This algorithm is independent of initialization, while the traditional clustering methods are dependent on initialization.

— *Approaches for graph clustering in heterogenous networks:* Heterogenous networks have multiple objects and link types. Homogenous networks can be derived from heterogenous networks. Heterogeneous networks carry richer information than their corresponding homogeneous networks. Mining semi structured heterogeneous networks helps us study clustering, ranking, classification, prediction, similarity search, and so on. Clustering and ranking are critical functions in data mining. Ranking will make sense within a particular cluster. The objects in the same cluster should not have same weights. High ranking objects in a cluster should be more important than the low ranking objects. Ranking as a feature is subject to a specific cluster. Clustering and ranking may mutually enhance each other. This is called ranking based clustering. Simple ranking considers immediate neighbors in a network. It is calculated as [189]

$$\vec{r}_X(x) = \frac{\Sigma_{j=1}^{n} W_{XY}(x,j)}{\Sigma_{i=1}^{m}\Sigma_{j=1}^{n} W_{XY}(i,j)}. \tag{75}$$

$$\vec{r}_Y(y) = \frac{\Sigma_{j=1}^{n} W_{XY}(i,y)}{\Sigma_{i=1}^{m}\Sigma_{j=1}^{n} W_{XY}(i,j)}. \tag{76}$$

Authority ranking is used when there is requirement of more sophisticated rules. The ranking scores are propagated in the different type networks. We discuss ranking rules by taking an example of the relationship between number of publications of an author and a conference [169].

— Rule 1: Highly ranked authors publish many papers in highly ranked conferences: $\vec{r}_Y(j) = \Sigma_{i=1}^{m} W_{YX}(j,i)\vec{r}_X(i)$. Here, $\vec{r}_Y(j)$ is the ranking of the author, $\Sigma_{i=1}^{m} W_{YX}(j,i)$ represents the venue of the conference and $\vec{r}_X(i)$ is the ranking of the venue.
— Rule 2: Numerous conferences are attracted from highly ranked conferences from many highly ranked authors: $\vec{r}_X = \Sigma_{j=1}^{n} W_{XY(i,j)\vec{r}_Y(j)}$.
— Rule 3: If any of the author is co-authors with other highly ranked authors, then the rank of an author is improved: $\vec{r}_Y(i) = \alpha\Sigma_{j=1}^{m} W_{YX}(i,j)\vec{r}_X(j) + (1-\alpha)\Sigma_{j=1}^{n} W_{YY}(i,j)\vec{r}_Y(j)$.

Other ranking functions are also used, e.g., using domain knowledge, i.e., journals may weigh more than conferences in the field of science.

From conditional rank distribution to new ranking measures: Considering a bi-typed bibliographic network, conditional ranking can be used to further improve clustering results. Conditional rank distribution can be viewed as a cluster feature. Considering every cluster X_k, the conditional rank scores, $r_X \mid X_K$ and $r_Y \mid X_k$ can be given as conditional rank distributions of X and Y (the features for cluster X_k). Cluster membership can be viewed as an object feature. It is proportional to its ranking in the clusters given by $p(k \mid x_i) \propto p(x_i \mid k)p(k)$. If its conditional rank in a cluster is high, possibility of an object to belong to that cluster $[(p(k \mid x_i))]$ increases. The attribute objects that are highly ranked have larger impact on determining the membership of a cluster in which it is targeted. Parameter estimation can be done using EM algorithm. In the E-step, calculate the distribution $p(z = k \mid y_j, x_i, \ominus)$ based on the current value of $\ominus^0$. The M step $\ominus$ is updated according to current $\ominus^0$.

Various approaches for clustering in heterogeneous networks are discussed as follows:

(1) RankClus (Integrated Clustering and Ranking in Heterogenous Networks) [169]: It is an EM style algorithm. At the beginning there is a simple network and then clusters are finalized based on ranking. Then the weights are adjusted, and the process is repeated. Initialization can be a random partition into K clusters. The objects are ranked in each sub network derived from each cluster and to estimate each target, object's mixture model coefficients are computed. The quality of the cluster can be validated using NMI (Normalized Mutual information) for N clusters. Let n (i,j) be the count of objects with cluster label i in the 1st clustering result and with cluster j in the 2nd clustering result. NMI is given by

$$\frac{\Sigma_{i=1}^{k}\Sigma_{j=1}^{k} p(i,j)\log\left(\frac{p(i,j)}{p_1(j)p_2(i)}\right)}{\sqrt{\Sigma_{j=1}^{k} p_1(j)\log p_1(j)\Sigma_{i=1}^{k} p_2(i)\log p_2(i)}}. \tag{77}$$

This method is an efficient way of cluster estimation as it follows the EM method and there are only linear computations.

(2) NetClus (Ranking-Based Clustering with Star Network Schema) [171]: Besides bi-typed network, this algorithm captures more semantics with multiple types of networks. Split a network into multi-subnetworks each of which is a multi-typed net cluster. The algorithm's foremost step is generating the target objects' initial partitions and inducing first net-clusters from the original network. We start a loop like an EM framework and repeat the following steps until the clusters do not change significantly.
— Building each net cluster's ranking based probabilistic generative model.
— Calculating each target object's posterior probabilities.
— Adjusting cluster assignment in accordance with the each cluster's posterior probabilities defined new measure.
To compute each attribute object's posterior probability within every net cluster.

(3) PathSim [168]: Similarity search is very important in cluster analysis. An important concept to understand is a Meta-Path. It is a meta-level description of a path amidst two objects present on network schema. It denotes a relationship that is already existing or is concatenated amidst two object types. Different meta-paths may depict different semantics. This is a popularly used measure is random walk (RW) [49]. The probability of a random walk starting at x and ending at y, with meta-path P can be given by

$$s(x, y) = \sum_{p \in P} prob(p). \tag{78}$$

There could be many concrete paths but we consider meta-walks. It is used in personalized PageRank (P-Pagerank) [72]. Random walk favors objects that are highly visible (large degree objects). The probability of Pairwise Random Walk (PRW) starting at (x,y) and ending at a common object (say z), following a meta-path (P_1, P_2) is given by

$$s(x, y) = \sum_{(p_1, p_2) \in (P_1, P_2)} prob(p_1) prob(p_2). \tag{79}$$

This similarity is also used in the SimRank. It favors pure objects (skewed distributed objects either in their in-links or out-links). SimRank and P-PageRank do not differentiate object type and relationship type. PathSim measure favors peers (objects having strong connectivity as well as similar visibility with a given meta-path (P).) Similarity of x and y is defined by

$$s(x, y) = \frac{2X \mid \{p_{x \rightsquigarrow y} : p_{x \rightsquigarrow y} \in P\} \mid}{\mid \{p_{x \rightsquigarrow x} : p_{x \rightsquigarrow x} \in P\} \mid + \mid \{p_{y \rightsquigarrow y} : p_{y \rightsquigarrow y} \in P\} \mid}. \tag{80}$$

Here, $x \rightsquigarrow y$ represents a meta-path. Meta-path based similarity computation can be costly. The overall cost can be reduced by storing short path schemes matrices and computing top-k queries on line. In order to do so, co-clusters are generated for materialized commuting matrices for target objects and feature objects. Then upper bounds are derived for objects and target cluster similarity as well as amidst object to object similarity. Then objects and target clusters are pruned securely if the current threshold is more than the upper bound similarity. Last, the top k threshold are updated in a dynamic manner. This is an efficient framework for implementing PathSim in a large dataset.

(4) User guided meta-path selection for clustering in heterogeneous networks: Different users may like to get different clusters for different clustering goals and this leads to choice of different meta-paths. For example, users can be asked to give seeds for desired clustering. The research problem becomes that we want user guided clustering with meta-path selection. The input for such type of clustering is
—The clustering T's target type
—K clusters
—Some clusters' seeds: $L_1, \ldots, L_k$
—Candidate meta-paths: $P_1, \ldots, P_M$
The expected output is
—Weight of each meta-path: $W_1, \ldots, W_m$
—Clustering results consistent along with the user guidance [166].
PathSelClus (A Probabilistic Modeling Approach) [170]: This approach is based on the basic idea of path selection clustering. It consists of three parts:
Part 1: Modeling and relationship generation
Part 2: Guidance and modeling from users
Part 3: Modeling the quality weights for meta-paths
The output differences based on the different meta-paths may have very subtle or very large differences. The important thing to note is that, the choice of the meta-path should be based on the data and not on personal preference.

A comprehensive comparison of the basic clustering approaches is given in Table 3.

3.2.11 Hadoop and Big Data. Apache Hadoop [180] is an open-source tool written in Java that facilitates distributed processing of large datasets across clusters of commodity hardwares. The most significant features of the Hadoop environment are HDFS and MapReduce programming framework. The framework is deployed in such a way that storage system is not physically separate from the processing system. The components of Hadoop ecosystem are shown in Figure 27. Various tasks are divided among these components. Data ingestion is done by Sqoop (structured data) and Flume (semi-structured and unstructured data); data is stored in HDFS, and processed using MapReduce or Spark. Yarn acts as the cluster resource management tool. Once the data is processed, it is analyzed using Pig, Hive, or Impala. HBASE works on top of HDFS and helps to analyze real-time data. In addition to this oozie is present as a work flow system. In order to query

Table 3. A Comprehensive Comparison of Clustering Approaches for Big Data

Approach	Proposed by	Model	Experimental Results					
			Capability of tackling large datasets	Sensitive to outliner/ noise	Time Complexity	Space Complexity	Unique Functional Properties	Experimental Highlights
Hierarchical approach								
Single linkage	[160]	Static	No	Yes	$O(n^2 \log n)$	$O(n^2)$	Represented as: $D(C_l, (C_i, C_j)) = min(D(C_l, C_i), D(C_l, C_j))$	–
Average linkage	[128]	Static	No	Yes	$O(n^2)$	$O(n^2)$	1. Is more complex agglomerative approach. 2. Includes average, median, centroid linkage and wards method	–
Complete linkage	[161]	Static	No	No	Obvious Algorithm: $O(n^3)$ Priority Queues: $O(n^2 \log n)$ Euclidean Plane: $O(n \log^2 n)$ Approximation Algorithm: $O(n \log n + n \log 2(1/\epsilon)$	–, $O(n^2)$, $O(n)$, $O(n)$	1. Represented as: $D(C_l, (C_i, C_j)) = max(D(C_l, C_i), D(C_l, C_j))$. 2. Forms spherical shapes and show reversal phenomenon	–
Hierarchical clustering algorithms								
BIRCH	[187]	Dynamic	Yes	No	$O(n)$	–	1. The important information is captured effectively while reducing the storage required doing the same. 2. This algorithm successfully eliminates the outliers.	1. It is a multilevel clustering approach. 2. The first phase constructs CF trees in memory. 3. The second phase clusters data.
CURE	[76]	Static	Yes	No	$O(n^2 logn)$	$O(n)$	1. It does not make any assumptions about the shape of clusters. 2. The distribution can be in the form of rings, S-shapes or strange bends. 3. Clusters are represented by a group of representation points instead of centroid.	1. It is a multilevel clustering approach. 2. The first phase constructs CF trees in memory. 3. The second phase clusters data.

(Continued)

Table 3. Continued

Approach	Proposed by	Model	Experimental Results					
			Capability of tackling large datasets	Sensitive to outliner/ noise	Time Complexity	Space Complexity	Unique Functional Properties	Experimental Highlights
CHAMELEON	[97]	Dynamic	Yes	Yes	$O(n(\log_2 n + m)$	–	1. It is based on nearest neighbor graph. 2. It uses minimal edge cut. .	1. It is a graph-based, two-phased algorithm. 2. The objects are partitioned into subgraphs by using a graph partitioning and then they are combined using hierarchical clustering.
K-Means approach								
K-Means clustering	[117]	Static	No	Yes	$(O(NKd))$	$O(N + K)$	Works best with compact and hyper spherical cluster shapes.	1. Value of number of clusters is assumed. 2. Binary search makes finding the correct value of k easier. 3. The brute force method is very costly.
BFR	[109]	Dynamic	Yes	No	–	–	It makes an assumption about the shape of clusters that they must be normally distributed about a centroid.	1. The data points are read in chunks. 2. The data in the memory is divided into discard, compressed and retained sets.
K means via PCA	[53]	Dynamic	Yes	No	–	–	Principal components are continuous solutions to discrete cluster membership indicators for K-means clustering.	1. Unsupervised dimension reduction is closely related to unsupervised learning. 2. New bounds of K-means clustering are within 0.5 - 1.5% of the optimal values.
ORCLUS	[8]	–	Yes	–	$O(K_o^3 + K_o N d + K_o^2 d^3)$	$O(K_o d^2)$		1. It begins with a set of K_o seeds that are randomly selected and have full dimensionality. 2. The dimensionality and the number of clusters keep on reducing till the predefined value of clusters is reached.

(Continued)

Table 3. Continued

			Experimental Results					
Approach	**Proposed by**	**Model**	**Capability of tackling large datasets**	**Sensitive to outliner/ noise**	**Time Complexity**	**Space Complexity**	**Unique Functional Properties**	**Experimental Highlights**
Random sampling approach								
CURE	[76]	Dynamic	Yes	No	$O(n^2 logn)$	$O(n)$	The initial clusters are formed using a standard clustering approach. Representative points are selected to merge clusters.	1. It does not make any assumptions about the shape of clusters. 2. The distribution can be in the form of rings, S-shapes or strange bends. 3. Clusters are represented by a group of representation points instead of centroid.
CLARA	[98]	Dynamic	Yes	No	$O(K(40+K)^2 + K(N-K))^+)$	–	Similar to CURE	Forms clusters with a medoid
Randomized search approach								
CLARANS	[133]	Dynamic	Yes	Yes	O(n)	–	Search can be visualized as being equivalent to traversing a graph.	1. Search Process in the form of a graph where each node corresponds to a set of *k* metoids. 2. Search can be visualized.
Condensation based approach								
BIRCH	[?]	Dynamic	Yes	No	$O(n)$	–	A new data structure clustering feature tree is used.	1. The important information is captured effectively while reducing the storage required doing the same. 2. This algorithm successfully eliminates the outliers.
Density based approach								
DBSCAN	[64]	Dynamic	Yes	No	$O(NlogN)$	$O(n^2)$	Based on the principle that if an object is a member of a cluster, then its neighborhood density should be reasonably high.	1. A data object creates new data objects by absorbing all objects in its neighborhood. 2. The data structure used is R* -tree structure.

(Continued)

Table 3. Continued

Approach	Proposed by	Model	Experimental Results					
			Capability of tackling large datasets	Sensitive to outliner/ noise	Time Complexity	Space Complexity	Unique Functional Properties	Experimental Highlights
DENCLUE	[159]	Dynamic	Yes	No	$O(NlogN)$	$O(n^2)$	Influences the data objects in the data space corresponding to it.	Generates clusters that are created by taking the maximum value of the density function.
Grid based approach								
WaveCluster	[159]	Dynamic	Yes	No	$O(N)$	–	It maps objects to frequency domain.	Defines a set of units in the given feature space and assigns data objects to these units.
Fractal Clustering	[22]	Dynamic	Yes	No	$O(N)$	–	There is a predefined condition of cluster's fractal dimension.	It has a predefined condition that fractal dimension of the cluster should be kept relatively stable.
STING	[177]	Dynamic	Yes	No	$O(NlogN)$	–	It works on the principle of forming a statistical grid.	It divides the spatial area into rectangular cells at different levels of resolution and these cells form a tree structure.
CLIQUE	[11]	Dynamic	Yes	No	$O(N)$(for number of objects), $O(n^2)$ for no. of dimensions	–	Generates clusters that are created by taking the maximum value of the density function.	1. It is based on both density and grid subspace clustering. 2. It defines subspace clusters as dense cells set in arbitrary subpsaces. 3. Quality of clusters is dependent on the appropriate selection of the number as well as width of the partitions and grid cells.
OptiGrid	[87]	Dynamic	Yes	No	Between $O(Nd)$ and $O(Nd \log N)$	–		1. Designed to get a prime grid-partitioning. 2. Achieved by creating cutting hyperplanes through a set of projections.

(Continued)

Table 3. Continued

Approach	Proposed by	Model	Experimental Results					
			Capability of tackling large datasets	Sensitive to outliner/ noise	Time Complexity	Space Complexity	Unique Functional Properties	Experimental Highlights
Graph based approach								
FADE	[149]	Static	Yes	No	O(e + n log n),where n and e are the numbers of nodes and edges	–	It uses decomposition trees to represent hierarchical clustering of the nodes.	1. It uses edges and multilevel visual abstraction. 2. decomposition tree provides a systematic way to determine the degree of closeness between nodes without explicitly calculating the distance between each node.
Spectral Clustering	[132]	Static	Yes	No	O(n3)	–	The data points are embedded in low dimensional space where clusters are more obvious.	1. This type of clustering makes no assumption on the shapes of clusters and can handle intertwined spirals etc. 2. It uses graph Laplacian's eigenvectors.
Ng-Jordon-Weiss(NJW) algorithm	[112]	Static	Yes	No	O(n2)	–	The method is well suited to the problem of appearance-based, on-line topological mapping for mobile robots.	1. Spectral clustering algorithm is applied to the affinity matrix after each row/column is added - which makes it possible to inspect the clusters as new data points are added. 2. only a fraction of the entries in the affinity matrix are computed.
Improved min-max cut graph clustering with nonnegative relaxation	[135]	Static	Yes	No	O(n2)	–	With the explicit nonnegative constraint, the proposed solutions are very close to the ideal class indicator matrix and can directly assign clusters to data points.	1. The authors apply additional nonnegative constraint into MinMax Cut graph clustering. 2. The algorithm always converges and significantly outperforms the traditional spectral relaxation approach.
Spectral Embedded Clustering	[139]	Static	Yes	No	–	–	In this framework a linearity regularization is explicitly added into the objective function of Spectral Clustering methods.	1. The proposed framework can naturally deal with out-of-sample data. 2. A new Laplacian matrix is constructed ,from a local regression, of each pattern, and incorporate it into the framework to capture both local and global discriminative information for clustering.

(Continued)

Table 3. Continued

Approach	Proposed by	Model	Experimental Results					
			Capability of tackling large datasets	**Sensitive to outliner/ noise**	**Time Complexity**	**Space Complexity**	**Unique Functional Properties**	**Experimental Highlights**
Constrained Laplacian Rank Algorithm for Graph-Based Clustering	[137]	Dynamic	Yes	No	–	–	The data graph itself is adjusted as part of the clustering procedure.	1. Two versions of this method are developed, based upon the L1-norm and the L2-norm, giving two new graph based clustering objectives. 2. Optimization algorithms are used to solve these objectives.
Initialization independent clustering with actively self-training method	[138]	Dynamic	Yes	No	–	–	This method is free of initialization, while traditional clustering methods are usually dependent on initialization.	1. It is an actively self training clustering method. 2. The framework is developed on graph to perform semi supervised learning, where Bayes error can be efficiently estimated.
Randomized greedy modularity clustering algorithm and the core group graph clustering scheme	[73]	Static	Yes	No	O(n)	–	It is a nondeterministic agglomerative hierarchical clustering approach which finds locally optimal solutions.	1. It is an ensemble learning clustering method which combines the local solutions of several base algorithms to form a good start solution. 2. The similarity of the randomized greedy modularity algorithm with incomplete solvers for the final algorithm is analyzed.
A partitioning-based divisive clustering technique for maximizing the modularity	[38]	Static	Yes	No	$O(\|V\|K + \|E\|)$ for a K-way clustering of a graph with $\|V\|$ vertices and $\|E\|$ edges.	–	The algorithm pursues a divisive clustering approach and uses established graph partitioning algorithms and techniques to compute recursive bi-partitions of the input as well as to refine clusters.	The algorithm accepts a weighted graph $G = (V, E, \omega)$, and returns the number of clusters K^* and the clustering $C^* = \{C1^*, C2^*, \ldots, CK^{**}\}$. It uses a bisection heuristic to compute a clustering of the given graph. Initially, all the vertices are in a single cluster.

(Continued)

Table 3. Continued

			Experimental Results					
Approach	**Proposed by**	**Model**	**Capability of tackling large datasets**	**Sensitive to outliner/ noise**	**Time Complexity**	**Space Complexity**	**Unique Functional Properties**	**Experimental Highlights**
Scalable and Accurate Graph Clustering and Community Structure Detection	[56]	Static	Yes	No	$O(n \log n)$.	–	The problem of finding a partition maximizing the modularity of a given graph G, can be reduced to a minimum weighted cut.	–
RankClus	[169]	Dynamic	Yes	No	$O(t_1\|E\| + t_2(K\|E\| + k + mK) + mK^2))$, t is iteration of whole algorithm, t_2 is iteration of mixture model, m is distance between each object, |E| is no. of links.	–	The problem of generating clusters for a specified type of objects, as well as ranking information for all types of objects based on these clusters in a multi- typed (i.e., heterogeneous) information network is addressed.	1. RankClus can generate more accurate clusters and in a more efficient way than the state-of-the-art link-based clustering methods. 2. Clustering results with ranks can provide more informative views of data compared with traditional clustering.
NetClus	[171]	Dynamic	Yes	No	$O(kmn)$	$O(mn)$	A loop like an EM framework is started and steps are repeated until the clusters do not change significantly.	1. The algorithm's foremost step is generating the target objects' initial partitions and inducing first net-clusters from original network. 2. Further, NetClus generates informative clusters, presenting good ranking and cluster membership information for each attribute object in each net-cluster.
PathSim	[168]	Dynamic	Yes	No	$((KN^2d^2)$, K is no. of iterations, N is total number of objects and d is average neighbor size.	$O(n^2)$	Based on the principle that similarity search that is defined among the same type of objects in heterogeneous networks.	1. It is an efficient solution that partially materializes short meta-paths and then concatenates them online to compute top-k results. 2. It is able to find peer objects in the network.

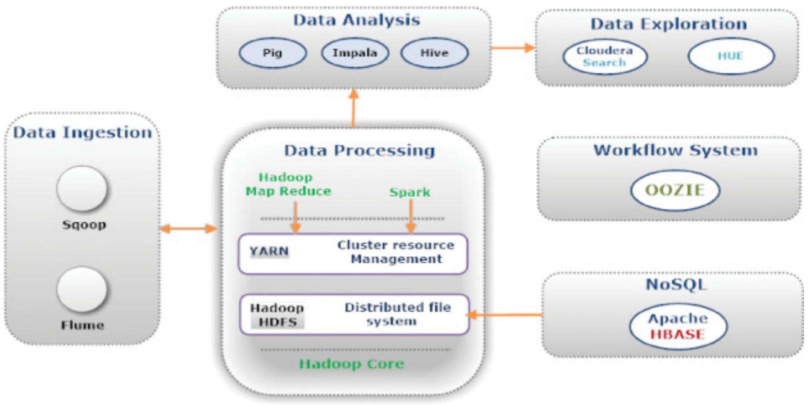


Fig. 27. Hadoop ecosystem - components.

data, there are user friendly interfaces such as cloudera search and Hue. MapReduce is the programming model used in Hadoop. Proposed by a Dean and Ghemawat at Google [50], this model is composed of two parts, known as mappers and reducers. At the high level, the mappers read the data from HDFS, process it and generate some intermediate results to the reducers. In the second phase, reducers are used to aggregate intermediate results to generate the final output which is again written to HDFS. In a typical Hadoop job, it is required to run several mappers and reducers across different nodes in the cluster [108]. Recently, there have been attempts to improve the efficiency of MapReduce in order to facilitate implementation of real world tasks such as search engine and machine learning. An example of this is Map-Reduce-Merge [183]. It adds a merge phase that efficiently merges data already partitioned and sorted by map and reduce modules. This new model can express relational algebra operations as well as implement join algorithms. As the volume of data keeps increasing it keeps becoming harder to cluster it. Many researchers have tried to design efficient parallel clustering algorithms. The authors in [188] have proposed a parallel k-means clustering algorithm based on MapReduce. The authors in [111] have implemented a parallel Apriori algorithm based on MapReduce. The results obtained indicate that algorithm scales well and efficiently process large datasets. The authors in [173] have proposed an improved K-means clustering algorithm to conduct clustering analysis based on medical data employing MapReduce computing framework. The authors in [157] have presented k-medoids clustering algorithm based on MapReduce paradigm to be able to perform clustering on large-scale data. The algorithm has achieved parallelism independent of the number of K clusters to be formed. The authors in [83] have proposed an approach for heavily skewed data. They have used MapReduce-based DBSCAN algorithm and it has proved to be scalable. There have also been efforts to develop big data benchmarks in order to efficiently analyze data and derive meaningful results. These benchmarks are designed keeping in mind the four v's of big data: volume, veracity, velocity, and variety [144]. There are a few benchmarks that use Hadoop as the software stacks. Some of them are as follows: Hibench [89] developed to perform on offline analytics, and real-time analytics. The applications that can be associated with it are sort, word-count, tersort, page-rank, Bayes-classification, nutch-indexing, and so on. GridMix [124] is used for online services to perform applications, like sort and sampling. Performance benchmark [146] is also used for online services with wider range of applications such as data loading, analyzing URL links, and so on. BigBench [176] is another benchmark that is used for offline analytics and real-time analytics. It can perform applications

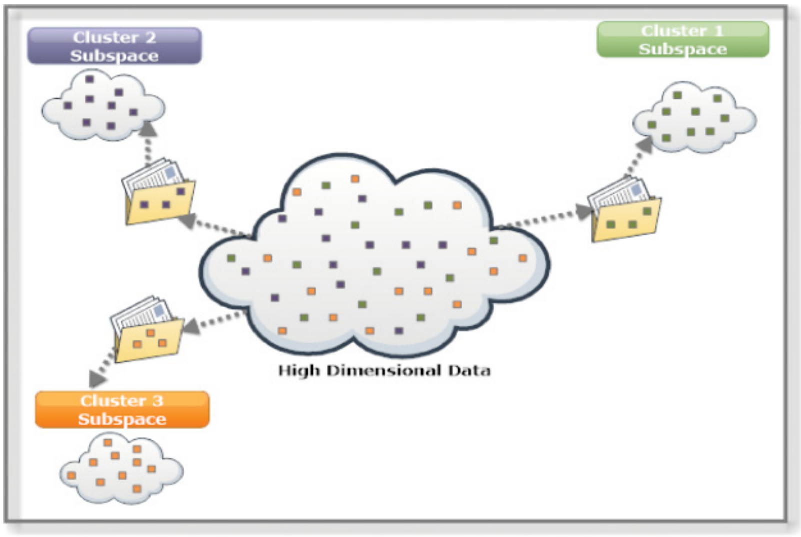


Fig. 28. General aim of clustering algorithm for high-dimensional datasets.

such as search engines, social networks, connected components, e-commerce collaborative, naive Bayes, relational database query analysis.

4 CLUSTERING HIGH-DIMENSIONAL DATASETS

Most of the algorithms discussed in the previous section, deal with one to three dimensions of data. Most of the results are obtained by deploying these approaches in two-dimensional data. They may not work well when number of dimensions grow to say 20. Many applications, such as text documents, or DNA microarray data may need to handle tens of thousands of dimensions of data. Cluster formulation in high-dimensional data is a problematic task, as there are a high number of irrelevant features of the data objects. The problem further complicates, when false correlations emerge, not only among the main features, but also among the subsets of these features. The biggest challenge arises while determining the relevancy of features according to a given cluster, in case feature selection is being used to form clusters. Additionally, different correlations among attributes may be relevant for different clusters. This phenomenon is known as feature relevance or local feature correlation. The most common way of overcoming these challenges is to perform dimensionality reduction, and then apply data mining techniques such as clustering on the reduced datasets. There are various approaches for dimensionality reduction such as Principal Component Analysis (PCA) that might be deployed. These methods map the original high-dimensional data space to a lower dimensional data space so as to form better clusters. However, approaches, such as these, are not suitable for all clustering problems. They are global in nature, which means that they compute an entire data space as a single subspace. While, in local feature correlation, multiple subspaces need to be evaluated as there might be different subspace for each cluster. Hence, general aim of clustering algorithms designed for high-dimensional data is to find clusters in arbitrarily oriented subspaces of the original data space. The search space of all possible subspaces housing clusters is restricted, but still exists in $O(2^d)$ [104]. The aim of clustering algorithms for high-dimensional datasets is illustrated in Figure 28. In this section, we discuss problem of cluster formulation in high-dimensional data and approaches used to cluster such data. Figure 29 shows the scope of this section.

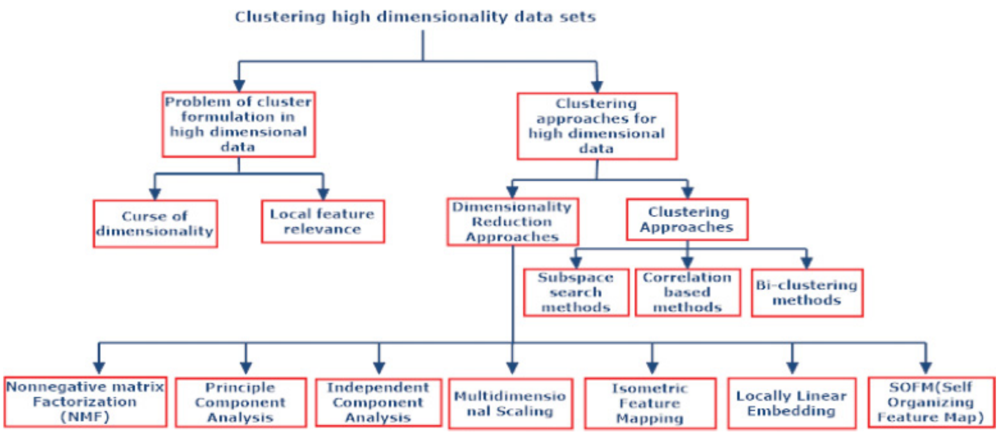

Clustering high dimensionality data sets
Problem of cluster formulation in high dimensional data
Clustering approaches for high dimensional data
Curse of dimensionality
Local feature relevance
Dimensionality Reduction Approaches
Clustering Approaches
Subspace search methods
Correlation based methods
Bi-clustering methods
Nonnegative matrix Factorization (NMF)
Principle Component Analysis
Independent Component Analysis
Multidimensional Scaling
Isometric Feature Mapping
Locally Linear Embedding
SOFM(Self Organizing Feature Map)

Fig. 29. Clustering high-dimensional datasets.

4.1 Problem of Cluster Formulation in High-dimensional Data

The "curse of dimensionality" states that there is an exponential increase in complexity of relationships as the dimensions keep on increasing [153]. This can be used to understand the problem faced in high-dimensional data spaces [25]. As the space dimensionality increases, there is decrease in the difference between the distance of data points. This means that the clustering distance measures, will no longer remain of any consequence in the high-dimensional spaces. As a result high-dimensional spaces cannot be clustered by distance-based algorithms. There have been attempts to overcome this problem. Recently, the authors in [136] have proposed an approach that learns data similarity matrix and clustering structure simultaneously. Data similarity matrix is learnt by assigning adaptive, and optimal neighbors for each data point based on local distances. This model has been extended to new clustering model for the projected clustering to handle high-dimensional data. The authors in [88] have tried to address the problem of curse of dimensionality. They have used joint dimensionality reduction and clustering to propose a discriminative embedded clustering framework. Within this framework, it is possible to view traditional approaches to reveal intrinsic relationships, it can also be simulated to develop a new method. In spite of such efforts, high-dimensional data analysis still remains a major challenge. There are five different aspects to the curse of dimensionality [63]:

(1) Optimization: The difficulty of global optimization increases exponentially with an increase in the number of dimensions.
(2) Distance concentration effect of the L_p-norms: This means that far and close neighbors have similar distances due to high-dimensionality. The relative contrast of L_p distances diminishes, as dimensionality increases.
(3) Irrelevant attributes: As discussed above, irrelevant attributes interfere with the performance of clustering for a particular object. There is difference in relevance of certain attributes to disparate object groups.
(4) Correlated attributes: Strong correlations among a subset of attributes can be used to reduce dimensionality. A dataset's intrinsic dimensionality can be considerably lesser than embedded dimensionality (i.e., the number of feature of the dataset).
(5) Data sparsity: Data volume in high-dimensional space is extremely sparse.

In practice, there is an exception to this curse. The intrinsic dimensionality in the many of the high-dimensional data spaces is quite lower than the original dimension [46]. This principle of dimensionality reduction makes high-dimensional data scalable, and provides a clear picture of the data under consideration. The downside of reduction of dimensionality is that there is loss of information that causes faulty result interpretation and even tends to distort real clusters. The most natural solution to this problem is identification of important data in the original dataset. This data can then be extracted to form meaningful clusters.

4.1.1 Clustering Approaches for High-dimensional Data. These approaches can be grouped in two categories [18]:

(1) Clustering approaches: Clusters may exist in more than one subspaces. There is a need to locate them in all these subspaces, starting from a low subspace (one-dimensional) to as far as we go. When data is projected in different subspaces, different clusters are formed. Subspaces might be axis parallel (parallel with some axis) or might be arbitrarily oriented. For finding clusters in axis parallel subspaces, bottom up or top down approaches may be used. While in arbitrarily oriented subspaces, correlation based clustering methods like ORCLUS are employed. There are a bi-clustering methods as well that are either optimization based or enumeration based [145].
(2) Dimensionality reduction approaches: A much lower dimensional space is constructed and clusters are searched there. There are chances that new dimensions can be constructed when some dimensions in the original data are combined. Spectral clustering and various dimensionality reduction methods fall in this category.

4.2 Clustering Approaches

Algorithms dealing with high-dimensional data have evolved over time. Traditional algorithms like K-means, hierarchical clustering, BFR, CURE, and so on [143] are the basic approaches used to find clusters in big data. Over the years, a lot of work has been done in this area and new hybrid approaches have been developed. In this section, we briefly discuss clustering approaches dealing with high-dimensional data.

— Subspace clustering: Subspace clustering focuses on the integral feature space by finding clusters in all subspaces. In CLIQUE [12], the entire data space is partitioned into equal sized units by using an axis parallel grid and units containing specified number of points is considered as dense. The maximum set of dense points adjacent to each other is termed as a cluster. Dense units satisfy the downward closure property. There is another variant of CLIQUE, known as ENCLUS [44] that instead of dense units, searches for subspaces containing one or more clusters relying on a fixed grid only. MAFIA [127] adopts the property of adaptive grid with subspace cluster generation, and is similar to CLIQUE. In nCluster [114], one- dimensional grid unit is formed by allowing the overlap of windows of δ length. SUBCLUE [96] searches in the density based clusters with the resulting clusters exhibiting the arbitrary shape or size in the corresponding subspaces. It computes the clusters by employing DBSCAN cluster model [64] to a particular subspace, but requires high runtime. DUSC [19] has dimensionality unbiased cluster model having drawback that it lacks search space pruning and anti-monotonic properties.
— Subspace search methods: Subspaces of a given data space are searched for finding clusters. The methods can be divided into two categories [105], top down and bottom up approaches. It usually starts from low-dimensional subspaces and moves up to higher dimensional subspaces. The top down approach is followed when there is an indication of existence of

clusters in higher dimensions. For the reduction in the searched high-dimensional subspaces, various pruning techniques are used, e.g., in case of CLIQUE [11]. It starts from full space and smaller subspaces are searched recursively. It is effective only if the assumption regarding locality is held that means the cluster subspaces are restricted, this can be determined by local neighborhood like is done in PROCLUS [6].

— Correlation based methods: Correlation clustering is a form of subspace clustering. It represents the most intuitive form of clustering construction. It gives solutions that can be approximated, while automatically selecting the number of clusters. This approach handles scenarios where the focus is on relationships between the objects, instead of on actual representations of the objects. The suitability of this method extends to the structured objects for which, feature vectors are not easy to obtain. Given the increasing scale of data these days, correlation clustering has become a powerful addition to the fields of data mining and agnostic learning. The task is to find a clustering that either maximizes agreements or minimizes disagreements. Many methodologies, such as approximations and linear programming formulations, have been used to approach this problem. Correlation clustering algorithms integrate dimensionality reduction algorithms with clustering approaches. We categorize the correlation clustering algorithms by the dimensionality reduction approaches they are based on [104]
  — PCA based approaches: Majority of correlation clustering approaches are based on this type of approach. HICO [3] is a hierarchical approach that uses "local correlation dimensionality" to define distance between data points. It also calculates the subspace orientation of the data points and uses "hierarchical density based clustering" in order to derive hierarchy of clusters. ORCLUS [7] picks k seeds first and assigns data objects to these seeds. The assignment is based on a distance function, based on an Eigenvector system. This approach chooses the highest value of k, hence increasing the efficiency of the algorithm. The local dimensionality reduction technique finds correlations among data objects in the locally correlated clusters [39]. This approach enhances multi-dimensional indexing, though it is a variant of ORCLUS. Another closely related variant has been proposed in [110]. It is more efficient than ORCLUS as it works on producing higher quality correlation clusters from noisy data. The algorithm 4C [29] is based on density of the data points. The number of clusters is not decided in advance, rather clusters expand around a seed until the density criteria is fulfilled. The density criteria specifies a minimum number of required points that should lie within the defined neighborhood of points. The definition of neighborhood comes from the distance matrix generated from the Eigen systems of two data points. COPAC [2] is a similar to 4C, but it improves some of the ambiguities in 4C. It partitions the data space such that the search is limited to finding clusters having data points demonstrating equal "local correlation dimensionality." This approach helps to reduce the time complexity rapidly to d^2. ERIC [1] is another approach based on Eigen values of a data point, derived from k-nearest neighbors. The neighborhood in this case is defined by approximate linear dependency of each data point. It also takes into consideration affine distance that is further defined by the strong eigenvectors of each point. The resulting clusters are ordered hierarchically to produce a hierarchy of subspace clusters. In [175], CURLER has been proposed. It is a method to detect arbitrary, non-linear correlations. It merges several clusters according to their co-sharing level. This makes it less sensitive to predefined number of clusters k.
  — Hough transform based approaches: Hough transform maps points form a 2-d space of Euclidean coordinates into a parameter space. This means that each point in data space

is associated with an infinite number of points, represented by a trigonometric function. The main feature of this transform is that the data points from the original data space are not kept under consideration. Objects may be far apart in the original data space but they are associated with a common line. The Hough transform does not require locality assumption and hence helps to obtain a global subspace clustering approach [126]. CASH uses grid-based methodology on the parameter space in order to carve out dense regions. It divides the data space based on attributes, and calculates the functions, that are intersecting the hyperboxes, formed as a result of the division of data space. If no correlation clusters are found in the original data space, then it can be inferred that there are no dense regions in the parameter space [40].

—Bi-Clustering methods: Bi-clustering is intended for clustering of attributes and objects simultaneously (treating attributes and objects in a symmetric way). Various requirements for finding bi-clusters are listed as
  —A few count of attributes are involved in a cluster.
  —There is participation of only small objects sets in a cluster.
  —There might be involvement of an attribute either in multiple clusters or not in even a single cluster.
  —There might be participation of an object either in multiple clusters or not in even a single cluster.

An example of bi-clustering is the microarray data or gene expression data. In a gene-sample/condition matrix, each element in a matrix consists of a real number for recording a gene's expression level under a particular condition [45]. The clustering of customer and products relationships can also be done by bi-clustering.

Types of Bi-clusters [119]: Let $A = \{a_1, \ldots, a_n\}$ be a set of elements in a matrix, $B = \{b_1, \ldots b_n\}$ be a set of conditions. A bi-cluster is a sub matrix where a consistent pattern is followed by elements and conditions. Then, there are four ideal cases of bi-cluster formulation for a submatrix I X J.

(1) Bi-clusters with constant values: For any i in I and j in J, $e_{ij} = c$. Here, c is a constant.
(2) Bi-clusters with constant values in rows: Here, $e_{ij} = c + \alpha_i$. It can also be constant values in columns.
(3) Bi-clusters with coherent values: Here, $e_{ij} = c + \alpha_i + \beta_j$.
(4) Bi-clusters with coherent evolutions in rows: $(e_{i1j1} - e_{i1i2})(e_{i2j1} - e_{i2j2}) \geq 0$, i.e., only having interest in down or up regulated changes across genes or conditions without having any constraint over the exact values.

4.3 Dimensionality Reduction Methods

Dimensionality reduction, as a concept, is required in instances where it is more useful to construct a new data space than to adopting the original data subspaces. For example, while clustering points in Figure 30, any original subspace projected on X- and Y-axes will not be clear as all of the three clusters being projected to X- and Y-axes will overlap. Upon constructing a different dimension such as the dashed one, the three clusters become more visible because the points are being projected onto a third dimension. Dimensionality reduction helps to reduce dimensionality by mathematical transformation. Various techniques for dimensionality reduction are discussed as follows [54]:

—Non-negative Matrix Factorization (NMF): One high-dimensional sparse non-negative matrix factorizes approximately into two low rank matrices. It works with the help of a group of algorithms in multivariate analysis, and linear algebra, where matrix "V" is factorized

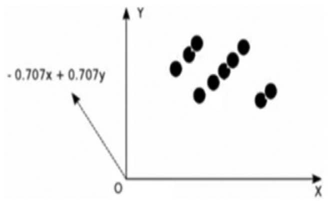


Fig. 30. Dimensionality reduction: An example [54].

into two matrices "W" and "H." The property of this matrix is that there are no negative elements. This property makes resulting matrices easier to inspect. NMF finds applications in fields such as computer vision, document clustering, chemometrics, audio signal processing, and recommender systems [172].

—Principle Component Analysis (PCA): It aims at constructing data vectors that indicate data variance in the given data in best possible way. It can be described as [21, 141]: An input matrix is $X = [x_1.......x_N]^T$ with linear mapping $Y = XV$ projects x_i to a low-dimensional subspace. Here, the resultant matrix is Y and projection matrix is V. This approach estimates projection matrix and aims to minimize the summation of error squares that occur in approximation of the input matrix. PCA proceeds by mapping input patterns into the feature space and aims to solve the eigenvalue problem by using feature value covariance matrix. It is suitable for Gaussian distributions since it is dependent on second order relationships formed in covariance matrix [60]. PCA is a mature technique that relies on the construction of a wide range of similarity measures that help to capture local correlations of attributes. This then helps to find arbitrarily oriented subspace clusters. The only drawback of this approach is the locality assumption. PCA is also very sensitive to outliers this means that if there are noise points in the local neighborhood then the subspace determination may be incorrect [192].

—Independent Component Analysis: It uses statistical information of higher order and is more suitable for non-Gaussian distribution [46]. It aims at finding the difference measures, i.e., locating components, that are independent from each other [90]. This problem can be simply formulated as $x = As$, where x is N-dimensional observable vector and s is L-dimensional, statistically independent, source vector. A in this case represents the NXL mixing matrix [184].

—Multidimensional Scaling: It aims at accommodating the original multivariate data into low-dimensional data space but keeping important proximity information safe. It is essentially a non-linear projection technique and measures distortion in data by using some criterion function [60, 184].

—Isometric Feature Mapping: It is based on Multidimensional Scaling. The shortest path amidst the points is calculated by the measurement of distances of input space. As it mostly uses Euclidean distance, it extends the applicability of Multidimensional Scaling to more complex non-linear structures in data [174].

—Locally Linear Embedding: It maintains that in the original data space, local relations are also operational in low-dimensional space in which it is projected. It is represented by a weight matrix, which describes the role of each point in the reconstruction of the related data points [23].

—SOFM (Self Organizing Feature Map): This clustering approach is based on a neural networks. It can represent high-dimensional input patterns in two-dimensional structure [101, 103]. A neuron is defined as each unit of the structure and a network of the input space is formed by connecting adjoining neurons. The neurons are all weight adjusted. It is more of a visual approach than a clustering approach [142]. A typical application is to cluster a massive set of documental data. The steps can be summarized as [102]

(1) Topology of SOFM is defined, and prototype vectors $m_i(0), i = 1 \ldots K$ are initialized in a random manner.
(2) The network is presented with input pattern x, and the node closest to x is selected. It is called the winning node (J):

$$J = \arg_i \min(||x - m_i||). \tag{81}$$

(3) Update the prototype vectors

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x - m_i(t)], \tag{82}$$

where $h_{ci}(t)$ is neighborhood function

$$h_{ci}(t) = \propto(t)\text{exp.}\left(-\frac{||r_c - r_i||^2}{2\sigma^2(t)}\right), \tag{83}$$

$\propto(t)$: To decrease the learning rate monotonically
r: Corresponding neuron's position
$\sigma(t)$: To decrease the kernel width function monotonically
Or, $h_{ci}(t) = \propto(t)$, if node c belongs to neighborhood of winning node J. Else, its 0.

(4) Steps 2 and 3 are repeated till the neurons start showing nothing more than a small positive number.

5 CHALLENGES AND LATEST TRENDS IN LARGE SCALE AND HIGH-DIMENSIONAL DATA CLUSTERING

As discussed in previous sections, handling large scale and high-dimensional data is a huge challenge. Data available for analysis these days is not always in structured form. Most of it is in semi-structured or unstructured forms like tweets, logs, hashtags, blogs, images, videos, and so on. Taking this data and deriving meaningful results out of it is the biggest challenge of data analysis. Existing data mining techniques, including the clustering, need to be revisited in order to make them compatible with distributed computing environments. Platforms like Hadoop and Spark have come up to support this kind of computation. Given the huge amount of data, it is humanly impossible to go over all the data points individually. Challenge here is to develop sophisticated machine learning models based on intelligent data mining techniques such as clustering that work in distributed environment to analyze and process the given data. Another issue that needs to be addressed is the quality of clusters. Clustering algorithms need to be sophisticated enough to adopt better feature selection approaches. Feature selection, if performed well can help to counter the drawbacks of dimensionality reduction. Another challenge in dealing with high-dimensional data is overlapping clustering. False correlations formulated in different subspaces lead to formulation of overlapping clustering that in turn results in false interpretation of data. Data visualization is another point of concern and research, as most of it is done by using graph theory, and clustering can play a huge role in making it more accurate. The graph clustering algorithms need to be improved in order to properly visualize data with huge size and large dimensions. All these challenges are being addressed by researchers in both academia and industry as solutions to these problems will help understand hidden meanings in data in a much better way. We now present some of the latest advancements in the field of data clustering and the future research opportunities they present.

- Clustering data streams: Data streams are very popular in the age of big data as there is a lot of data coming from sensors, video feed, tweets, blogs, and so on that record data in the online data version. Data streams are continuous, ordered, changing, fast, and have huge volume. A very important algorithm to deal with this type of data is single-scan algorithm. There are multiple data streams flowing in a stream processing system. The users/applications are continuously sending queries to this system. The stream processing system, taking the multiple streams tries to return query results in real time [75]. The challenge is how can we perform cluster analysis in such fast changing data stream. There are some clustering approaches that have been developed to deal with this type of data:
  - A *K*-median approach [140]: It is *k*-median method based approach. Data stream points are from metric space, so medians can be easily computed. *K*-clusters are found in the stream to minimize the summation of distances from data points to their closest centers. As it deals with a data stream, it works as a constant factor approximation algorithm. It works as a simple two-step algorithm in small memory space. $O(k)$ centers in the string like $S_1, \ldots, S_i$ are searched by S_i for each set of M records. Each point in S_i is assigned to its closest center by performing local clustering. Let S' be center weighted by the number of points assigned to it. This means that the points and their weights are considered. Finally, cluster S' is used to find k centers. This is a typical approach to summarize data stream in real-time environment. It deals with an online process dealing with a large number of microclusters. Microclusters represent local density estimates by aggregating the information of many data points in a defined area [77].
  - Hierarchical clustering tree method [107]: When there are many data points, but there is no capacity to process all of them, then this method is used. In this approach at maximum level (m – i) medians are being maintained. At $O(k)$ level—(i + 1) medians of weights equal to the summation of the intermediate medians' weights assigned to them are developed on seeing m of them. A major concern is that quality will suffer for evolving data streams (maintaining only m level—i median) and the functionality is limited in exploring clusters over various portions of streams over time.
- CluStream Approach [4]: The design goals of this algorithm include getting a clustering of high quality evolving data streams with stream mining and rich functionality. Stream mining consists of only one pass over the stream data with restricted space usage but carrying high efficacy. This clustering methodology has three major points.
  1. Tilted time frame work: Most recent data is stored, so that dynamic changes are not lost.
  2. Microclustering: Data is stored in a fine clustering scale having superior quality than *k*-means/*k*-median. It is incremental, where online processing is facilitated and cluster maintenance is ensured.
  3. Two stage processing: Microclustering (for maintaining online data streams and its history) and macroclustering (to answer ad-hoc queries). This will ensure that with limited overhead, scalability; high efficiency; power of evolution/change; detection and quality of results can be achieved.

  Following concepts play an important role in understanding CluStream approach and are also potential areas of research in the field of data clustering.
  - Time dimension (A tilted time model) [43]: A tilted time frame is a trade off between space and granularity. It decides if snapshots of the statistical information are stored at what moment. The general design has natural (follow the natural time-frame like quarterly, weekly, monthly, etc.), logarithmic (jumps in logarithmic way, like $t, t, 2t, 4t, 8t, 16t, 32, 64t, \ldots$) and pyramidal (designed by clustering) tilted time frames.

- Pyramidal tilted timeframe: It has been adopted by CluStream. It can be explained by taking an example. Six frames can be considered where each accepts a maximum of three snapshots. For a snapshot number N, if N mod $2^d = 0$, it is inserted to frame numbered d. The oldest snapshot is eliminated if there are more than three. The pyramidal pattern is followed in storing the snapshots of microclusters set. They are accumulated at different granularity levels, which is dependent on the recency. The classification of snapshots is done into different orders varying from 1 to $\log(T)$. The snapshots of ith order appear at intervals of α^i with $\alpha \geq 1$. The last $(\alpha + 1)$ snapshots are only stored.

The CluStream framework is a microclustering approach that uses BIRCH CF-Tree structure. A height balanced tree for storing the CFs is known as CF tree. The sums of the children of CFs are stored in non-leaf nodes. Since all the points are not stored in the tree, the leaf nodes store microclusters. The data locality's statistical information is contained in these. The temporal extension of the cluster-feature vector $\overline{X}_1 \ldots \overline{X}_k \ldots$ is in it because it contains the multidimensional points with time stamps $T_1 \ldots T_k \ldots$. The d dimensions, i.e., $\overline{x}_t = (x_i^1 \ldots x_i^d)$ are contained by each point. The n points microcluster is termed as a $(2d + 3)$ tuple. Keeping in mind the dynamic nature of data these days, CluStream divides clustering process into two components:

1. Online components: These are essentially microcluster maintenance. When the data stream is received, its statistical summary is periodically stored. Q microclusters are initially created. Here, Q is generally significantly larger than the natural clusters count. When the online incremental updates come, then it is checked if new point is within max-boundary and then the point is inserted into microcluster. Otherwise, a new cluster is created. Obsolete microclusters may be deleted or two closest clusters might be merged.
2. Offline components: This is a query-based macroclustering. When the user query is received, they are answered based on stored summary statistics. Based on a user specific time horizon h and the count of macroclusters k, the computation of macroclusters is done by employing k-means or some other clustering algorithms. Quality clusters with minimal overhead are generated.

- Ensemble clustering [121]: It tries to combine many clustering models' results to create a more robust clustering. Since clustering is unsupervised, the optimal clustering is captured by no single model or criterion, but a more robust solution is provided by an ensemble clustering of models. The methodology for the same can be described as
  - Generate k different clustering (i.e., ensemble components): The selection of the ensemble components can be model based (combine different parameters of different models) or data selection based (select data from different samples of data).
  - The different results are combined into a single or more robust solution. This can be done by two major methods:
    1. Hypergraph partitioning: Each data point is considered as a vertex and a cluster in any of the ensemble components is represented as a hyper-edge.
    2. Meta-clustering: It is a graph-based approach, except for the reason that in the ensemble components, vertices are associated with each cluster.

6 APPLICATIONS OF LARGE SCALE AND HIGH-DIMENSIONAL DATA CLUSTERING

In this section, we introduce some interesting applications of data clustering:

- Name disambiguation [130]: It is one task in data cleaning performed by link-based clustering. In this case, object reconciliation is done instead of object distinction. Object reconciliation is recognizing that different objects/people share the same names, while object distinction says that same name is assigned to different objects. The challenge in performing object distinction is that sexual similarity cannot be detected, since they have the exact same name. In this scenario, link analysis may take advantage of redundancy to facilitate entity cross checking and validation. An example of such an approach is an algorithm that applies object distinction by information network-based cluster analysis [116]. Another way to solve this challenge is to find neighborhood similarity. The similarity between their contexts is indicated by neighbor tuples of each reference. A major problem is to identify the boundaries to cut the links to get distinct clusters. The most widely used approach is to use self-boosting in order to train the same bulky dataset to set the boundaries. In order to train with same dataset, we use following steps:

- Build a training set automatically by following steps:
    1. Select distinct names.
    2. Some similarity is shared by the collaboration behavior within the same community.
    3. The parameters are trained by employing a common and large set of unambiguous examples.
- A model is learned using SVM for combination of various join paths. It involves following steps:
- Each join path is used as two attributes (with neighborhood similarity and link-based similarity).
- The model is all attributes' weighted sum.

Data clustering is also done by measuring similarity between clusters. There are different cases that can be used to measure similarity. We compare them to see which one of them is the best for name disambiguation:

- Single link: This means getting highest similarity between points. Using this is not an optimum solution as there is an easy connection between different objects' references.
- Complete link: There exists minimal similarity among them. It is also not an optimum solution as there may be a weak connection between reference to the same object.
- Average link: There exists average similarity amidst points in two clusters. It is a superior measure. A refinement over it is average neighborhood similarity and collective random walk probability.

- Evolution of heterogenous networks [9]: Networks evolve with time, they make sequences based on patterns. The motivation is to model evolution of communities in heterogeneous networks. The focus is to automatically detect the best number of communities in each timestamp, then model the smoothness between communities of adjacent timestamps. The evolution structure has to be modeled explicitly like birth, death, and split of subnetworks. An example of this application is an algorithm called the EvoNetClus [167]. It is clustering for dynamic heterogenous networks. We see co-evolution within a community. This means that within the cluster, there are heterogenous multi-types objects/links. This approach discovers evolution structures among different communities. The general idea is to find evolutionary communities from network sequences. A graphical model represents heterogenous networks in the best possible way. It is a generative model based on Dirichlet process mixture model [163]. At each timestamp, a cluster is dependent on historical cluster and background cluster distribution. Figure 31 represents a generative graphic model. Here, each graph is generated using a lot of parameters and then it is transformed to the next graph by applying some transformative operations on it, i.e., $G_1 \rightarrow G_2 \rightarrow \cdots \rightarrow G_t$. To generate a new object O_i. It is decided whether to join an existing community or a new one.

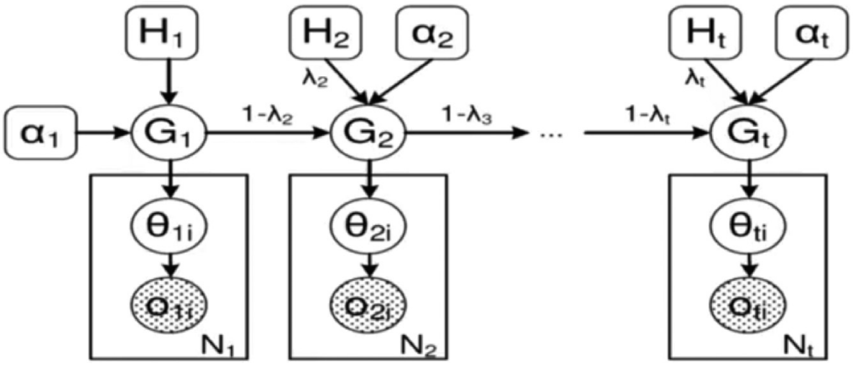


Fig. 31. A generative graphic model.

Join an existing community k with probability $n_k/(l-1+\alpha)$. To join a new community k with probability $\alpha/(l-1+\alpha)$, its prior is decided. It is based on either from background distribution (λ) or historical communities ($(1-\lambda)\pi_k$), with different probabilities, and draw the attribute distribution from the prior. o_i is generated according to attribute distribution:

$$p(o_{i,t} \mid z_{i,t} = k, \Theta_t) = p(o_{i,t} \mid \theta_{k,t}) = p(a_{i,t} \mid \theta^A_{k,t})p(c_{i,t} \mid \theta^C_{k,t})p(d_{i,t} \mid \theta^D_{k,t})$$
$$= \Pi^{|A|}_{j=1}\theta^A_{k,t}(j)^{a_{ij,t}}\Pi^{|C|}_{j=1}\theta^C_{k,t}(j)^{c_{ij,t}}\Pi^{|D|}_{j=1}\theta^D_{k,t}(j)^{d_{ij,t}}.$$

Greedy inference for each timestamp uses Collapse Gibbs sampling, which is trying to sample cluster label for each target object. Based on this, we can understand community evolution discovery algorithm

—Customer segmentation and collaborative filtering [113]: Cluster segmentation is clustering of customer behavior on the similarities of their profiles and shopping behaviors. While collaborative filtering and recommendation systems are based on customer ratings of products.

—Text mining [24]: It uses cluster analysis a lot, as it involves co-clustering of words and documents. There have been many recent studies on how to do clustering, typing, profiling, and in depth analysis of large collection of text/web documents.

—Clustering of multimedia and social media [129]: It is widely used for summarizing multimedia data (e.g., videos) and integrated analysis of news and tweets.

—Temporal and sequence applications [95]: This area has applications in wide variety of domains like clustering and web log click streams to find patterns of users and reorganize web page structures or clustering biological sequence data to construct biological models and discover anomalies.

—Social network analysis [156]: Clustering analysis is used for finding hidden social communities, i.e., community detection. It is also used for subsequent analysis for link prediction, influence analysis, classification, and anomaly detection.

Apart from these, there are many data mining and broad social applications. This is an emerging field and all kinds of approaches and applications are being proposed and developed everyday. Figure 32 represents applications discussed above and the clustering approaches that can be used to implement them.

7 CONCLUSION AND FUTURE SCOPE

Data clustering is a tool of data mining, and is much sought after approach in the field of unsupervised learning. Clustering process involves selecting the data on which cluster analysis is to be

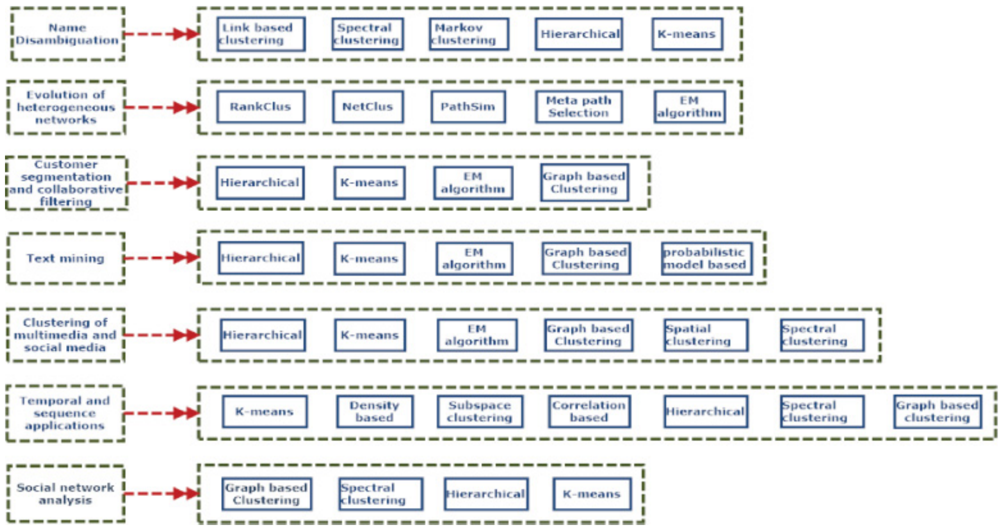


Fig. 32. Clustering approaches that can be used by a particular application.

performed, process that data and use a clustering approach to analyze it. Each of the steps of the clustering process is linked with a feedback mechanism, making it an iteratively ongoing process, which can be modified according to need. Once the appropriate data is obtained, the next question arises as to which similarity measure is suitable for use in the current situation. Also, we need to ask that in what way can the domain knowledge be utilized in the given situation. In the end, we select a suitable approach to analyze the given dataset. Complex present day problems cannot be solved by a single clustering algorithm. This means that no approach is best in itself. Its suitability depends on the problem at hand. Due to advancement in technology and computational ability, we now have the capability to process large-scale datasets and high-dimensional datasets. This survey provides a systematic review of the clustering process, approaches, challenges, latest trends, and applications of large and high-dimensional datasets. It is a comprehensive document that describes the fundamental concepts of clustering, such as clustering tendency and validation, and explains the clustering process. The main focus of this survey is to present clustering approaches for big data and high-dimensional data. A comprehensive comparison of these approaches is also presented. In order to make it more relevant for the current times, latest trends, challenges, and applications are also discussed.

As future work, there are many research topics that can be considered. Ensemble clustering can be further explored to see if ensembles of single clustering algorithms are more stable and accurate than individual clustering. Further, ensemble of multi-clustering algorithms and ensemble of a single clustering should be compared to see which one is more stable and accurate. Distributed computing is the future of data analysis, the future of clustering is to incorporate the concept of distributed system with clustering, in order to improve performance and efficiency of existing algorithms. A sub-problem associated with this challenge is to find the most appropriate parameter settings for each of these new distributed computing algorithms.

REFERENCES

[1] Elke Achtert, Christian Bohm, Hans-Peter Kriegel, Peer Kroger, and Arthur Zimek. 2007b. On exploring complex relationships of correlation clusters. In *Proceedings of the 19th International Conference on Scientific and Statistical Database Management (SSBDM'07)*. IEEE, 7–7.

[2] Elke Achtert, Christian Böhm, Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. 2007a. Robust, complete, and efficient correlation clustering. In *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 413–418.

[3] Elke Achtert, Christian Bohm, Peer Kroger, and Arthur Zimek. 2006. Mining hierarchies of correlation clusters. In *Proceedings of the 18th International Conference on Scientific and Statistical Database Management (SSDBM'06)*. IEEE, 119–128.

[4] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. 2003. A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases-Volume 29 (VLDB'03)*. 81–92.

[5] Charu C. Aggarwal and S. Yu Philip. 2004. A condensation approach to privacy preserving data mining. In *Proceedings of the International Conference on Extending Database Technology, Advances in Database Technology (EDBT'04)*. Springer, 183–199.

[6] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. 1999. Fast algorithms for projected clustering. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, Vol. 28. ACM, 61–72.

[7] Charu C. Aggarwal and Philip S. Yu. 2000. *Finding Generalized Projected Clusters in High Dimensional Spaces*, Vol. 29. ACM.

[8] Charu C. Aggarwal and Philip S. Yu. 2002. Redefining clustering for high-dimensional applications. *IEEE Transactions on Knowledge and Data Engineering* 14, 2 (2002), 210–225.

[9] Charu C. Aggarwal and ChengXiang Zhai. 2012a. *Mining Text Data*. Springer Science & Business Media.

[10] Charu C. Aggarwal and ChengXiang Zhai. 2012b. A survey of text clustering algorithms. In *Mining Text Data*. Springer, 77–128.

[11] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. 1998. *Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications*, Vol. 27. ACM.

[12] Rakesh Agrawal, Johannes Ernst Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. 1999. Automatic subspace clustering of high dimensional data for data mining applications. U.S. Patent 6,003,029, issued December 14, 1999.

[13] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval* 12, 4 (2009), 461–486.

[14] Amineh Amini, Teh Ying Wah, and Hadi Saboohi. 2014. On density-based data streams clustering algorithms: a survey. *Journal of Computer Science and Technology* 29, 1 (2014), 116–141.

[15] Rajaraman Anand and D. U. Jeffrey. 2012. *Mining of Massive Datasets*.

[16] S. Aranganayagi and K. Thangavel. 2007. Clustering categorical data using silhouette coefficient as a relocating measure. In *Proceedings of the 2007 International Conference on Computational Intelligence and Multimedia Applications*, Vol. 2. IEEE, 13–17.

[17] Saurabh Arora and Inderveer Chana. 2014. A survey of clustering techniques for big data analysis. In *Proceedings of the 2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*. IEEE, 59–65.

[18] Ira Assent. 2012. Clustering high dimensional data. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2, 4 (2012), 340–350.

[19] Ira Assent, Ralph Krieger, Emmanuel Muller, and Thomas Seidl. 2007. DUSC: Dimensionality unbiased subspace clustering. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM'07)*. IEEE, 409–414.

[20] Abdelkarim Ben Ayed, Mohamed Ben Halima, and Adel M. Alimi. 2014. Survey on clustering methods: Towards fuzzy clustering for big data. In *Proceedings of the 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR'14)*. IEEE, 331–336.

[21] Pierre Baldi and Kurt Hornik. 1989. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks* 2, 1 (1989), 53–58.

[22] Daniel Barbará and Ping Chen. 2000. Using the fractal dimension to cluster datasets. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 260–264.

[23] Mikhail Belkin and Partha Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS'01)*, Vol. 14. 585–591.

[24] Michael W. Berry and Malu Castellanos. 2004. Survey of text mining. *Computing Reviews* 45, 9 (2004), 548.

[25] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. 1999. When is nearest neighbor meaningful? In *Proceedings of the International Conference on Database Theory (ICDT'99)*. Springer, 217–235.

[26] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. 2000. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 7 (2000), 719–725.

[27] Christopher M. Bishop. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press.

[28] Leon Bobrowski and James C. Bezdek. 1991. c-means clustering with the l l and l norms. *IEEE Transactions on Systems, Man and Cybernetics* 21, 3 (1991), 545–554.

[29] Christian Böhm, Karin Kailing, Peer Kröger, and Arthur Zimek. 2004. Computing clusters of correlation connected objects. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. ACM, 455–466.

[30] Urszula Boryczka. 2009. Finding groups in data: Cluster analysis with ants. *Applied Soft Computing* 9, 1 (2009), 61–70.

[31] Olutayo Boyinbode, Hanh Le, and Makoto Takizawa. 2011. A survey on clustering algorithms for wireless sensor networks. *International Journal of Space-Based and Situated Computing* 1, 2–3 (2011), 130–136.

[32] Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. 2003. Experiments on graph clustering algorithms. In *Proceedings of the European Symposium on Algorithms*. Springer, 568–579.

[33] Janez Brank, Marko Grobelnik, and Dunja Mladenic. 2005. A survey of ontology evaluation techniques. In *Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD'05)*. 166–170.

[34] Ryan P. Browne, Paul D. McNicholas, and Matthew D. Sparling. 2012. Model-based learning using a mixture of mixtures of Gaussian and uniform distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 4 (2012), 814–817.

[35] Peter Brucker. 1978. On the complexity of clustering problems. In *Optimization and Operations Research*. Springer, 45–54.

[36] Joachim Buhmann. 1995. Data clustering and learning. In *The Handbook of Brain Theory and Neural Networks*, Michael A. Arbib (Ed.). MIT Press, 278–281.

[37] Gail A. Carpenter, Stephen Grossberg, and David B. Rosen. 1991. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks* 4, 6 (1991), 759–771.

[38] Umit V. Catalyiirek, Kamer Kaya, Johannes Langguth, and Bora Uçar. 2013. A partitioning-based divisive clustering technique for maximizing the modularity. *Graph Partitioning and Graph Clustering* 588 (2013), 171.

[39] Kaushik Chakrabarti and Sharad Mehrotra. 2000. Local dimensionality reduction: A new approach to indexing high dimensional spaces. In *Proceedings of the 26th VLDB Conference*. 89–100.

[40] Asis Kumar Chattopadhyay, Tanuka Chattyopadhyay, Tuli De, and Saptarshi Mondal. 2013. Independent component analysis for dimension reduction classification: Hough transform and CASH algorithm. In *Astrostatistical Challenges for the New Astronomy*. Springer, 185–202.

[41] C. L. Philip Chen and Chun-Yang Zhang. 2014. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences* 275 (2014), 314–347.

[42] Min Chen, Shiwen Mao, and Yunhao Liu. 2014. Big data: A survey. *Mobile Networks and Applications* 19, 2 (2014), 171–209.

[43] Yixin Chen, Guozhu Dong, Jiawei Han, Benjamin W. Wah, and Jianyong Wang. 2002. Multi-dimensional regression analysis of time-series data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*. 323–334.

[44] Chun-Hung Cheng, Ada Waichee Fu, and Yi Zhang. 1999. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 84–93.

[45] Yizong Cheng and George M. Church. 2000. Biclustering of expression data. *ISMB* 8 (2000), 93–103.

[46] Vladimir Cherkassky and Filip M. Mulier. 2007. *Learning from Data: Concepts, Theory, and Methods*. John Wiley & Sons.

[47] Michael Cochez and Hao Mou. 2015. Twister tries: Approximate hierarchical agglomerative clustering for average distance in linear time. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 505–517.

[48] Ronan Collobert and Samy Bengio. 2001. SVMTorch: Support vector machines for large-scale regression problems. *The Journal of Machine Learning Research* 1 (2001), 143–160.

[49] Nick Craswell and Martin Szummer. 2007. Random walks on the click graph. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 239–246.

[50] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified data processing on large clusters. *Communications of the ACM* 51, 1 (2008), 107–113.

[51] Hongbo Deng and Jiawei Han. 2013. Probabilistic models for clustering. In *Data Clustering: Algorithms and Applications*, Charu C. Aggarwal and Chandan K. Reddy (Eds.). CRC Press, 61.

[52] Inderjit S. Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 269–274.

[53] Chris Ding and Xiaofeng He. 2004. K-means clustering via principal component analysis. In *Proceedings of the 21st International Conference on Machine Learning*. ACM, 29.

[54] Chris Ding, Xiaofeng He, Hongyuan Zha, and Horst D. Simon. 2002. Adaptive dimension reduction for clustering high dimensional data. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02).* IEEE, 147–154.

[55] Chris H. Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D. Simon. 2001. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'01).* IEEE, 107–114.

[56] Hristo N. Djidjev and Melih Onus. 2013. Scalable and accurate graph clustering and community structure detection. *IEEE Transactions on Parallel and Distributed Systems* 24, 5 (2013), 1022–1029.

[57] Chuong B. Do and Serafim Batzoglou. 2008. What is the expectation maximization algorithm? *Nature Biotechnology* 26, 8 (2008), 897–899.

[58] Richard C. Dubes. 1993. Cluster analysis and related issues. In *Handbook of Pattern Recognition & Computer Vision*, C. H. Chen, L. F. Pau, and P. S. P. Wang (Eds.). World Scientific Publishing Co., Inc., 3–32.

[59] Jordi Duch and Alex Arenas. 2005. Community detection in complex networks using extremal optimization. *Physical Review E* 72, 2 (2005), 027104.

[60] Richard O. Duda, Peter E. Hart, and David G. Stork. 2001. *Pattern Classification* (2nd ed.). Wiley.

[61] Jack Edmonds. 1965. Paths, trees, and flowers. *Canadian Journal of Mathematics* 17, 3 (1965), 449–467.

[62] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. 1998. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America* 95, 25 (1998), 14863–14868.

[63] Stefano Ermon, Carla Gomes, Ashish Sabharwal, and Bart Selman. 2013. Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML'13)*, 334–342.

[64] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, Vol. 96. 226–231.

[65] Brian Everitt and Torsten Hothorn. 2011. Cluster analysis. In *An Introduction to Applied Multivariate Analysis with R*, Robert Gentleman, Kurt Hornik, and Giovanni Parmigiani (Eds.). Springer, 163–200.

[66] Adil Fahad, Najlaa Alshatri, Zahir Tari, Abdullah Alamri, Ibrahim Khalil, Albert Y. Zomaya, Sebti Foufou, and Abdelaziz Bouras. 2014. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing* 2, 3 (2014), 267–279.

[67] Gary William Flake, Robert E. Tarjan, and Kostas Tsioutsiouliklis. 2004. Graph clustering and minimum cut trees. *Internet Mathematics* 1, 4 (2004), 385–408.

[68] Chris Fraley and Adrian E. Raftery. 1998. How many clusters? Which clustering method? Answers via model-based cluster analysis. *The Computer Journal* 41, 8 (1998), 578–588.

[69] Laurent Galluccio, Olivier Michel, Pierre Comon, Mark Kliger, and Alfred O. Hero. 2013. Clustering with a new distance measure based on a dual-rooted tree. *Information Sciences* 251 (2013), 96–113.

[70] Laurent Galluccio, Olivier Michel, Pierre Comon, Mark Kliger, and Alfred O. Hero. 2013. Hybrid clustering algorithm with modifications enhanced K-means and hierarchal clustering. *International Journal of Advanced Research in Computer Science and Software Engineering* 3, 5 (2013), 166–170.

[71] Junhao Gan and Yufei Tao. 2015. DBSCAN revisited: Mis-claim, un-fixability, and approximation. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data.* ACM, 519–530.

[72] Esther Garcia, Francisco Pedroche, and Miguel Romance. 2013. On the localization of the personalized PageRank of complex networks. *Linear Algebra and Its Applications* 439, 3 (2013), 640–652.

[73] Andreas Geyer-Schulz and Michael Ovelgönne. 2014. The randomized greedy modularity clustering algorithm and the core groups graph clustering scheme. In *German-Japanese Interchange of Data Analysis Results*, Wolfgang Gaul, Andreas Geyer-Schulz, Yasumasa Baba, and Akinori Okada (Eds.). Springer, 17–36.

[74] K. Chidananda Gowda and Edwin Diday. 1991. Symbolic clustering using a new dissimilarity measure. *Pattern Recognition* 24, 6 (1991), 567–578.

[75] Sudipto Guha, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. 2000. Clustering data streams. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science.* IEEE, 359–366.

[76] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. 1998. CURE: An efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, Vol. 27. ACM, 73–84.

[77] Michael Hahsler and Matthew Bolaños. 2016. Clustering data streams based on shared density between micro-clusters. *IEEE Transactions on Knowledge and Data Engineering* 28, 6 (2016), 1449–1461.

[78] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. 2001. On clustering validation techniques. *Journal of Intelligent Information Systems* 17, 2 (2001), 107–145.

[79] Greg Hamerly and Charles Elkan. 2002. Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the 11th International Conference on Information and Knowledge Management.* ACM, 600–607.

[80] Jiawei Han, Micheline Kamber, and Jian Pei. 2011. *Data Mining: Concepts and Techniques.* Elsevier.

[81] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. 2015. The rise of big data on cloud computing: Review and open research issues. *Information Systems* 47 (2015), 98–115.

[82] Richard J. Hathaway, James C. Bezdek, and Yingkang Hu. 2000. Generalized fuzzy c-means clustering strategies using L p norm distances. *IEEE Transactions on Fuzzy Systems* 8, 5 (2000), 576–582.

[83] Yaobin He, Haoyu Tan, Wuman Luo, Shengzhong Feng, and Jianping Fan. 2014. MR-DBSCAN: A scalable MapReduce-based DBSCAN algorithm for heavily skewed data. *Frontiers of Computer Science* 8, 1 (2014), 83–99.

[84] Zengyou He, Xiaofei Xu, and Shengchun Deng. 2008. k-ANMI: A mutual information based clustering algorithm for categorical data. *Information Fusion* 9, 2 (2008), 223–233.

[85] Monika Henzinger. 2006. Finding near-duplicate web pages: A large-scale evaluation of algorithms. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 284–291.

[86] Alexander Hinneburg and Daniel A. Keim. 1998. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)*, Vol. 98. 58–65.

[87] Alexander Hinneburg and Daniel A. Keim. 1999. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proceedings of the 25th VLDB Conference.*

[88] Chenping Hou, Feiping Nie, Dongyun Yi, and Dacheng Tao. 2015. Discriminative embedded clustering: A framework for grouping high-dimensional data. *IEEE Transactions on Neural Networks and Learning Systems* 26, 6 (2015), 1287–1299.

[89] Shengsheng Huang, Jie Huang, Jinquan Dai, Tao Xie, and Bo Huang. 2010. The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. In *Proceedings of the 2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW'10).* IEEE, 41–51.

[90] Aapo Hyvarinen. 1999. Survey on independent component analysis. *Neural Computing Surveys* 2, 4 (1999), 94–128.

[91] Anil K. Jain and Richard C. Dubes. 1988. *Algorithms for Clustering Data.* Prentice-Hall, Inc.

[92] Anil K. Jain, Robert P. W. Duin, and Jianchang Mao. 2000. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 1 (2000), 4–37.

[93] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. 1999. Data clustering: A review. *ACM Computing Surveys* 31, 3 (1999), 264–323.

[94] Glen Jeh and Jennifer Widom. 2002. SimRank: A measure of structural-context similarity. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 538–543.

[95] Huidong Jin, Jie Chen, Hongxing He, Graham J. Williams, Chris Kelman, and Christine M. O'Keefe. 2008. Mining unexpected temporal associations: Applications in detecting adverse drug reactions. *IEEE Transactions on Information Technology in Biomedicine* 12, 4 (2008), 488–500.

[96] Karin Kailing, Hans-Peter Kriegel, and Peer Kröger. 2004. Density-connected subspace clustering for high-dimensional data. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, Vol. 4. SIAM.

[97] George Karypis, Eui-Hong Han, and Vipin Kumar. 1999. Chameleon: Hierarchical clustering using dynamic modeling. *Computer* 32, 8 (1999), 68–75.

[98] Leonard Kaufman and Peter J. Rousseeuw. 2009. *Finding Groups in Data: An Introduction to Cluster Analysis*, Vol. 344. John Wiley & Sons.

[99] Yoonsoo Kim and Mehran Mesbahi. 2006. On maximizing the second smallest eigenvalue of a state-dependent graph Laplacian. *IEEE Transactions on Automatic Control* 51, 1 (2006), 116–120.

[100] Jon Kleinberg. 2003. An impossibility theorem for clustering. In *Proceedings of the 15th International Conference on Neural Information Processing Systems.* 463–470.

[101] Teuvo Kohonen. 1990. The self-organizing map. *Proceedings of the IEEE* 78, 9 (1990), 1464–1480.

[102] Teuvo Kohonen, Samuel Kaski, Krista Lagus, Jarkko Salojärvi, Jukka Honkela, Vesa Paatero, and Antti Saarela. 2000. Self organization of a massive document collection. *IEEE Transactions on Neural Networks* 11, 3 (2000), 574–585.

[103] Teuvo Kohonen, M. R. Schroeder, and T. S. Huang. 2001. *Self-Organizing Maps.* Springer-Verlag, New York, Inc., Secaucus, NJ, 43.

[104] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. 2008. Detecting clusters in moderate-to-high dimensional data: Subspace clustering, pattern-based clustering, and correlation clustering. *Proceedings of the VLDB Endowment* 1, 2 (2008), 1528–1529.

[105] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. 2009. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data* 3, 1 (2009), 1.

[106] G. N. Lance and W. T. Williams. 1967. A general theory of classification sorting strategies: 1= hierarchical systems, 2= clustering systems. *Computer Journal* 10, 3 (1967), 271–277.

[107] Peter Langfelder, Bin Zhang, and Steve Horvath. 2008. Defining clusters from a hierarchical cluster tree: The dynamic tree cut package for R. *Bioinformatics* 24, 5 (2008), 719–720.

[108] Kyong-Ha Lee, Yoon-Joon Lee, Hyunsik Choi, Yon Dohn Chung, and Bongki Moon. 2012. Parallel data processing with MapReduce: A survey. *ACM SIGMOD Record* 40, 4 (2012), 11–20.

[109] Deyi Li, Shuliang Wang, Wenyan Gan, and Deren Li. 2012. Data field for hierarchical clustering. *Developments in Data Extraction, Management, and Analysis* (2012), 303.

[110] Jiuyong Li, Xiaodi Huang, Clinton Selke, and Jianming Yong. 2007. A fast algorithm for finding correlation clusters in noise data. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 639–647.

[111] Ning Li, Li Zeng, Qing He, and Zhongzhi Shi. 2012. Parallel implementation of Apriori algorithm based on MapReduce. In *Proceeding of the 2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD)*. IEEE, 236–241.

[112] Xin-Ye Li and Li-jie Guo. 2012. Constructing affinity matrix in spectral clustering based on neighbor propagation. *Neurocomputing* 97 (2012), 125–130.

[113] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.

[114] Bing Liu, Yiyuan Xia, and Philip S. Yu. 2000. Clustering through decision tree construction. In *Proceedings of the 9th International Conference on Information and Knowledge Management*. ACM, 20–29.

[115] Chung Laung Liu. 1968. *Introduction to Combinatorial Mathematics*, Vol. 181. McGraw-Hill, New York.

[116] Yuechang Liu and Yong Tang. 2015. Network based framework for author name disambiguation applications. *International Journal of u-and e-Service, Science and Technology* 8, 9 (2015), 75–82.

[117] Stuart P. Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137.

[118] James MacQueen and others. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1. Oakland, CA, 281–297.

[119] Sara C. Madeira and Arlindo L. Oliveira. 2004. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 1, 1 (2004), 24–45.

[120] Jianchang Mao and Anil K. Jain. 1996. A self-organizing network for hyperellipsoidal clustering (HEC). *IEEE Transactions on Neural Networks* 7, 1 (1996), 16–29.

[121] Marie-Hélène Masson and Thierry Denoeux. 2011. Ensemble clustering in the belief functions framework. *International Journal of Approximate Reasoning* 52, 1 (2011), 92–109.

[122] Geoffrey J. McLachlan and Kaye E. Basford. 1988. *Mixture Models: Inference and Applications to Clustering*. Statistics: Textbooks and Monographs. Dekker, New York, Dekker.

[123] Ryszard S. Michalski and Robert E. Stepp. 1983. Automated construction of classifications: Conceptual clustering versus numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 4 (1983), 396–410.

[124] Zijian Ming, Chunjie Luo, Wanling Gao, Rui Han, Qiang Yang, Lei Wang, and Jianfeng Zhan. 2013. BDGS: A scalable big data generator suite in big data benchmarking. In *Workshop on Big Data Benchmarks*, Tilmann Rabl, Nambiar Raghunath, Meikel Poess, Milind Bhandarkar, Hans-Arno Jacobsen, and Chaitanya Baru (Eds.). Springer, 138–154.

[125] Jiawei Han and Micheline Kamber. 2001. *Data Mining: Concepts and Techniques*. Elsevier.

[126] Priyanka Mukhopadhyay and Bidyut B. Chaudhuri. 2015. A survey of hough transform. *Pattern Recognition* 48, 3 (2015), 993–1010.

[127] T. M. Murali and Simon Kasif. 2003. Extracting conserved gene expression motifs from gene expression data. In *Pacific Symposium on Biocomputing*, Vol. 8. 77–88.

[128] Fionn Murtagh. 1983. A survey of recent advances in hierarchical clustering algorithms. *Computer Journal* 26, 4 (1983), 354–359.

[129] Mor Naaman. 2012. Social multimedia: Highlighting opportunities for search and mining of multimedia data in social media applications. *Multimedia Tools and Applications* 56, 1 (2012), 9–34.

[130] Mohammad Hossein Nadimi and Mostafa Mosakhani. 2015. A more accurate clustering method by using co-author social networks for author name disambiguation. *Journal of Computing and Security* 1, 4 (2015), 307–317.

[131] Mark E. J. Newman. 2004. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems* 38, 2 (2004), 321–330.

[132] Andrew Y. Ng, Michael I. Jordan, Yair Weiss, and others. 2001. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS'01)*, Vol. 14. 849–856.

[133] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*. Santiago, Chile, 144–155.

[134] Raymond T. Ng and Jiawei Han. 2002. CLARANS: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering* 14, 5 (2002), 1003–1016.

[135] Feiping Nie, Chris Ding, Dijun Luo, and Heng Huang. 2010. Improved minmax cut graph clustering with nonnegative relaxation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 451–466.

[136] Feiping Nie, Xiaoqian Wang, and Heng Huang. 2014. Clustering and projected clustering with adaptive neighbors. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 977–986.

[137] Feiping Nie, Xiaoqian Wang, Michael I. Jordan, and Heng Huang. 2016. The constrained Laplacian rank algorithm for graph-based clustering. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*. Citeseer, 1969–1976.

[138] Feiping Nie, Dong Xu, and Xuelong Li. 2012. Initialization independent clustering with actively self-training method. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42, 1 (2012), 17–27.

[139] Feiping Nie, Zinan Zeng, Ivor W. Tsang, Dong Xu, and Changshui Zhang. 2011. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Transactions on Neural Networks* 22, 11 (2011), 1796–1808.

[140] Liadan O'callaghan, Adam Meyerson, Rajeev Motwani, Nina Mishra, and Sudipto Guha. 2002. Streaming-data algorithms for high-quality clustering. In *Proceedings of the 18th International Conference on Data Engineering*. IEEE, 0685.

[141] Erkki Oja. 1992. Principal components, minor components, and linear neural networks. *Neural Networks* 5, 6 (1992), 927–935.

[142] Nikhil R. Pal, James C. Bezdek, and Eric C. K. Tsao. 1993. Generalized clustering networks and Kohonen's self-organizing scheme. *IEEE Transactions on Neural Networks* 4, 4 (1993), 549–557.

[143] Divya Pandove and Shivani Goel. 2015. A comprehensive study on clustering approaches for big data mining. In *Proceedings of the 2015 2nd International Conference on Electronics and Communication Systems (ICECS)*. IEEE, 1333–1338.

[144] Divya Pandove and Shivani Goel. 2015. Prototyping and in-depth analysis of big data benchmarking. In *Proceedings of the 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM)*. IEEE, 1222–1229.

[145] Lance Parsons, Ehtesham Haque, and Huan Liu. 2004. Subspace clustering for high dimensional data: A review. *ACM SIGKDD Explorations Newsletter* 6, 1 (2004), 90–105.

[146] Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, and Michael Stonebraker. 2009. A comparison of approaches to large-scale data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. ACM, 165–178.

[147] Adriano Pereira, Leonardo Rocha, Fernando Mourão, Paulo Góes, and Wagner Meira Jr. 2009. Reactivity based model to study online auctions dynamics. *Information Technology and Management* 10, 1 (2009), 21–37.

[148] K. Rajendra Prasad and B. Eswara Reddy. 2013. Assessment of clustering tendency through progressive random sampling and graph-based clustering results. In *Proceedings of the 2013 IEEE 3rd International Advance Computing Conference (IACC)*. IEEE, 726–731.

[149] Aaron Quigley and Peter Eades. 2000. FADE: Graph drawing, clustering, and visual abstraction. In *International Symposium on Graph Drawing*. Springer, 197–210.

[150] M. Kuchaki Rafsanjani, Z. Asghari Varzaneh, and N. Emami Chukanlo. 2012. A survey of hierarchical clustering algorithms. *The Journal of Mathematics and Computer Science* 5, 3 (2012), 229–240.

[151] Anand Rajaraman, Jeffrey D. Ullman, Jeffrey David Ullman, and Jeffrey David Ullman. 2012. *Mining of Massive Datasets*, Vol. 1. Cambridge University Press, Cambridge.

[152] I. K. Ravichandra Rao. 2003. Data mining and clustering techniques. In *Proceedings of DRTC Workshop on Semantic Web*, Vol. 8.

[153] Bellman Richard. 1961. *Adaptive Control Processes: A Guided Tour*. Princeton University Press.

[154] Andrew Rosenberg and Julia Hirschberg. 2007. V-Measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of EMNLP-CoNLL*, Vol. 7. 410–420.

[155] Satu Elisa Schaeffer. 2007. Graph clustering. *Computer Science Review* 1, 1 (2007), 27–64.

[156] John Scott. 2012. *Social Network Analysis*. Sage.
[157] M. Omair Shafiq and Eric Torunski. 2016. A parallel K-medoids algorithm for clustering based on MapReduce. In *Proceedings of 15th IEEE International Conference on Machine Learning and Applications (ICMLA'16)*. IEEE, 502–507.
[158] B. A. Shboul and Sung-Hyon Myaeng. 2009. Initializing k-means using genetic algorithms. (2009).
[159] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. 1998. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of VLDB*, Vol. 98. 428–439.
[160] Peter H. A. Sneath. 1957. The application of computers to taxonomy. *Microbiology* 17, 1 (1957), 201–226.
[161] Thorvald Sørensen. 1948. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. *Biologiske skrifter* 5 (1948), 1–34.
[162] Michael Steinbach, George Karypis, Vipin Kumar, and others. 2000. A comparison of document clustering techniques. In *Proceedings of KDD Workshop on Text Mining*, Vol. 400. Boston, 525–526.
[163] Mark Steyvers and Tom Griffiths. 2007. Probabilistic topic models. *Handbook of Latent Semantic Analysis* 427, 7 (2007), 424–440.
[164] Eric A. Stone and Julien F. Ayroles. 2009. Modulated modularity clustering as an exploratory tool for functional genomic inference. *PLoS Genetics* 5, 5 (2009), e1000479.
[165] Mu-Chun Su and Chien-Hsing Chou. 2001. A modified version of the K-means algorithm with a distance based on cluster symmetry. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 6 (2001), 674–680.
[166] Yizhou Sun and Jiawei Han. 2013. Meta-path-based search and mining in heterogeneous information networks. *Tsinghua Science and Technology* 18, 4 (2013), 329–338.
[167] Yizhou Sun and Jiawei Han. 2013. Mining heterogeneous information networks: A structural analysis approach. *ACM SIGKDD Explorations Newsletter* 14, 2 (2013), 20–28.
[168] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. In *Proceedings of the VLDB Endowment.* 11.
[169] Yizhou Sun, Jiawei Han, Peixiang Zhao, Zhijun Yin, Hong Cheng, and Tianyi Wu. 2009. Rankclus: Integrating clustering with ranking for heterogeneous information network analysis. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology.* ACM, 565–576.
[170] Yizhou Sun, Brandon Norick, Jiawei Han, Xifeng Yan, Philip S. Yu, and Xiao Yu. 2013. Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 7, 3 (2013), 11.
[171] Yizhou Sun, Yintao Yu, and Jiawei Han. 2009. Ranking-based clustering of heterogeneous information networks with star network schema. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 797–806.
[172] Rashish Tandon and Suvrit Sra. 2010. Sparse nonnegative matrix approximation: New formulations and algorithms. *Rapport Technique* 193 (2010), 38–42.
[173] Zhuo Tang, Kunkun Liu, Jinbo Xiao, Li Yang, and Zheng Xiao. 2017. A parallel k-means clustering algorithm based on redundance elimination and extreme points optimization employing MapReduce. *Concurrency and Computation: Practice and Experience* 29, 20 (2017), 1–18.
[174] Joshua B. Tenenbaum, Vin De Silva, and John C. Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290, 5500 (2000), 2319–2323.
[175] Anthony K. H. Tung, Xin Xu, and Beng Chin Ooi. 2005. Curler: Finding and visualizing nonlinear correlation clusters. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data.* ACM, 467–478.
[176] Lei Wang, Jianfeng Zhan, Chunjie Luo, Yuqing Zhu, Qiang Yang, Yongqiang He, Wanling Gao, Zhen Jia, Yingjie Shi, Shujie Zhang, and others. 2014. Bigdatabench: A big data benchmark suite from internet services. In *Proceedings of the IEEE 20th International Symposium on High Performance Computer Architecture (HPCA'14).* IEEE, 488–499.
[177] Wei Wang, Jiong Yang, Richard Muntz, and others. 1997. STING: A statistical information grid approach to spatial data mining. In *Proceedings of VLDB*, Vol. 97. 186–195.
[178] William J. Welch. 1982. Algorithmic complexity: Three NP-hard problems in computational statistics. *Journal of Statistical Computation and Simulation* 15, 1 (1982), 17–25.
[179] Douglas Brent West and others. 2001. *Introduction to Graph Theory*, Vol. 2. Prentice Hall Upper Saddle River.
[180] Tom White. 2012. *Hadoop: The Definitive Guide.* O'Reilly Media, Inc.
[181] Rui Xu and Donald Wunsch. 2005. Survey of clustering algorithms. *IEEE Transactions on Neural Networks* 16, 3 (2005), 645–678.
[182] Rui Xu, Donald Wunsch, and others. 2005. Survey of clustering algorithms. *IEEE Transactions on Neural Networks* 16, 3 (2005), 645–678.
[183] Hung-chih Yang, Ali Dasdan, Ruey-Lung Hsiao, and D. Stott Parker. 2007. Map-reduce-merge: Simplified relational data processing on large clusters. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data.* ACM, 1029–1040.

[184] Forrest W. Young. 2013. *Multidimensional Scaling: History, Theory, and Applications*. Psychology Press.

[185] Jane Yang Yu and Peter Han Joo Chong. 2005. A survey of clustering schemes for mobile ad hoc networks. *IEEE Communications Surveys & Tutorials* 7, 1 (2005), 32–48.

[186] Btissam Zerhari, Ayoub Ait Lahcen, and Salma Mouline. 2015. Big data clustering: Algorithms and challenges. In *Proceedings of the International Conference on Big Data, Cloud and Applications (BDCA'15).*

[187] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of ACM Sigmod Record*, Vol. 25. ACM, 103–114.

[188] Weizhong Zhao, Huifang Ma, and Qing He. 2009. Parallel k-means clustering based on MapReduce. In *Proceedings of IEEE International Conference on Cloud Computing*. Springer, 674–679.

[189] Ding Zhou, Sergey A. Orshanskiy, Hongyuan Zha, and C. Lee Giles. 2007. Co-ranking authors and documents in a heterogeneous network. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM'07).* IEEE, 739–744.

[190] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment* 2, 1 (2009), 718–729.

[191] Xinhua Zhuang, Yan Huang, Kannappan Palaniappan, and Yunxin Zhao. 1996. Gaussian mixture density modeling, decomposition, and applications. *IEEE Transactions on Image Processing* 5, 9 (1996), 1293–1302.

[192] Arthur Zimek. 2009. Correlation clustering. *ACM SIGKDD Explorations Newsletter* 11, 1 (2009), 53–54.

Received September 2016; revised June 2017; accepted August 2017