Chapter 2

Related Work

Cluster analysis is the process of division of data into groups (clusters) that are meaningful, useful, or both. It should reflect the natural structure of the data. Cluster analysis groups data objects based on information present in the data. The goal of clustering is to group similar or related objects in the same cluster and different or unrelated groups in different clusters. Clustering objects into meaningful groups is based on similarity or dissimilarity measures. Cluster analysis is a difficult problem because of many factors such as effective similarity measures, criterion functions, algorithms and initial conditions. Moreover, it is well known that no clustering method can adequately handle all sorts of cluster structures (shape, size and density). Outlier detection is one of the major technologies in data mining, whose task is to find small groups of data objects that are considerably different from rest of the data. Outlier mining is applied over various fields such as telecommunication, financial fraud detection, and data cleaning. In outlier mining, the patterns lying behind the outliers are usually interesting and helps the decision makers to make profit or to improve the service quality. It is considered to be an important research area, and outlier detection is studied intensively by the data mining community [BKNS00, HK06]. In the succeeding sections, we discuss some commonly used proximity measures, various well-known clustering techniques and also some of the existing outlier detection techniques are discussed.
2.1 Proximity Measures

From scientific and mathematical point of view, distance is defined as a quantitative degree of how far apart two objects are. Similarity is a numerical quantity that reflects the strength of relationship between two objects or two features. Similarities are higher for pairs of objects that are more alike. This quantity is usually in the range of either -1 to +1 or is normalized into 0 to 1. If the similarity between a pair of objects \((x, y)\) is denoted by \(S_{x,y}\), we can measure this quantity in several ways depending on the scale of measurement (or data type) that we have.

Distance measure is also known as dissimilarity measure. Similarity and dissimilarity measures are often called proximity measures. Dissimilarity measures the discrepancy between the two objects, i.e., it measures the degree to which two objects are different. There are many types of distance and similarity measures. Each similarity or dissimilarity measure has its own characteristics. Next, we consider several important issues concerning proximity measures.

2.1.1 Relationship between Similarity and Dissimilarity

Let normalized dissimilarity between object \(x\) and object \(y\) be denoted by \(d_{x,y}\). Then the relationship between dissimilarity and similarity [HK06] is given by

\[
S_{x,y} = 1 - d_{x,y}. \tag{2.1}
\]

Here, \(S_{x,y}\) is normalized similarity between objects \(x\) and \(y\). Similarity is bounded by 0 and 1. When similarity is one (i.e., two objects are exactly similar), the dissimilarity is zero and when the similarity is zero (i.e., two objects are very different), dissimilarity is one. If the value of similarity has range of -1 to +1, and the dissimilarity is measured with range of 0 and 1, then

\[
S_{x,y} = 1 - 2d_{x,y}. \tag{2.2}
\]

When dissimilarity is one (i.e., two objects are very different), similarity is minus one and when the dissimilarity is zero (i.e., two objects are very similar), similarity is one. In many cases, measuring dissimilarity (i.e., distance) is easier than measuring
similarity. Once we can measure dissimilarity, we can easily normalize it and convert it to similarity measure. It is also common for dissimilarities to range from 0 to $\infty$. Frequently, proximity measures are transformed to the interval $[0, 1]$. The transformation of similarities to the interval $[0, 1]$ is given by

$$S'_{x,y} = \frac{S_{x,y} - \text{min}_{x,y} S_{x,y}}{\text{max}_{x,y} S_{x,y} - \text{min}_{x,y} S_{x,y}}$$

(2.3)

where, $\text{min}_{x,y}$ and $\text{max}_{x,y}$ are minimum and maximum similarities respectively. Similarly, dissimilarity measures with a finite range can be mapped to the interval $[0, 1]$ by using the formula

$$d'_{x,y} = \frac{d_{x,y} - \text{min}_{x,y} d_{x,y}}{\text{max}_{x,y} d_{x,y} - \text{min}_{x,y} d_{x,y}}$$

(2.4)

where, $\text{min}_{x,y}$ and $\text{max}_{x,y}$ are minimum and maximum dissimilarities respectively.

If the proximity measure has values in the range $[0, \infty]$, then a non-linear transformation is needed and the values in the transformed scale will not have the same relationship to one another as the original. But, whether such a transformation is desirable or not depends on the application it is used.

### 2.1.2 Some Distance Measures

A popular distance measure based on variables that take on continuous values is to standardize the values by dividing by the standard deviation (sometimes other measures such as range are used) and then to compute the distance between objects using the Euclidean metric.

The *Euclidean distance* $d_{i,j}$ between two objects, $i$ and $j$ with variable values $(x_{i1}, x_{i2}, \ldots, x_{im})$ and $(x_{j1}, x_{j2}, \ldots, x_{jn})$ is defined by:

$$d_{i,j} = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{im} - x_{jn})^2}$$

(2.5)

If some variables should be given more importance than others then the squared difference terms should be multiplied by weights (positive numbers adding up to
one) and use larger weights for the important variables. The *Weighted Euclidean distance* measure is given by:

\[ d_{i,j} = \sqrt{w_1(x_{i1} - x_{j1})^2 + w_2(x_{i2} - x_{j2})^2 + \cdots + w_n(x_{in} - x_{jn})^2} \]  

(2.6)

where \( w_1, w_2, \cdots, w_n \) are the weights for variables 1, 2, \cdots, \( n \) so that \( w_i \geq 0, \sum_{i=1}^{n} w_i = 1 \).

Other useful measures of dissimilarity other than the Euclidean distance that satisfy the triangular inequality and so qualify as distance metrics are:

*Manhattan distance* is defined by

\[ d_{i,j} = \sum_{m=1}^{n} |x_{im} - x_{jm}| \]  

(2.7)

*Minkowski distance* is the generalized form of the two distance metrics discussed above. It is given as

\[ d_{i,j} = \left( \sum_{m=1}^{n} |x_{im} - x_{jm}|^p \right)^{1/p} \]  

(2.8)

where \( p \) is a parameter. For \( p = 1 \), we get the Manhattan distance, and for \( p = 2 \) we get the Euclidean distance.

*Mahalanobis distance* corrects data for different scales and correlations in the variables. It is defined by

\[ d_{i,j} = \sqrt{(x_i - x_j)'V^{-1}(x_i - x_j)} \]  

(2.9)

where \( x_i \) and \( x_j \) are \( n \)-dimensional vectors of the variable values for \( i \) and \( j \) respectively; and \( V \) is the covariance matrix for these vectors. This measure takes into account the correlation between the variable: variables that are highly correlated with other variables do not contribute as much as variables that are uncorrelated or mildly correlated.

*Maximum co-ordinate distance* is defined by

\[ d_{i,j} = \max_{m=1,2, \cdots, n} |x_{im} - x_{jm}| \]  

(2.10)
For $p = \infty$ we get the Chebyshev distance ($L_{\max}$ or $L_\infty$ norm) named after Chebyshev [HDRT04]. This is the maximum distance between any pair of attributes of the objects. Formally, $L_\infty$ is defined as

$$d_{i,j} = \lim_{p \to \infty} \left( \sum_{m=1}^{n} |x_{im} - x_{jm}|^p \right)^{\frac{1}{p}} . \quad (2.11)$$

**Hamming distance** [Ham50] between two strings of equal length is the number of positions at which the corresponding symbols are different. It measures the minimum number of substitutions required to change one member into another. For a fixed length $n$, the Hamming distance is a metric on the vector space of the words of that length, as it obviously fulfills the conditions of non-negativity, identity of indiscernibles and symmetry, and it can be shown easily by complete induction that it satisfies the triangle inequality as well. For binary strings $i$ and $j$ the Hamming distance is equal to the number of ones in $i$ XOR $j$. If $q_{i0} =$ number of variables with value 1 for the $i^{th}$ object and 0 for the $j^{th}$ object and $q_{j0} =$ number of variables with value 0 for the $i^{th}$ object and 1 for the $j^{th}$ object, we have

$$d_{i,j} = q_{i0} + q_{j0} \quad (2.12)$$

### 2.1.3 Some Similarity Measures

Sometimes it is more natural or convenient to work with a similarity measure between objects rather than distance which measures dissimilarity. Such measures can always be converted to distance measures. In the above example we could define a distance measure $d_{i,j} = 1 - S_{i,j}$.

**Pearson’s correlation**: The correlation coefficient, $\rho_{i,j}$ is a widely used similarity measure, defined by

$$\rho_{i,j} = \sqrt{\frac{\sum_{m=1}^{n}(x_{im} - \overline{x_i})(x_{jm} - \overline{x_j})}{\sqrt{\sum_{m=1}^{n}(x_{im} - \overline{x_i})^2 \sum_{m=1}^{n}(x_{jm} - \overline{x_j})^2}}} \quad (2.13)$$

where, $\overline{x_i}$ is the mean of the $n$ attributes of the $i^{th}$ object and $\overline{x_j}$ is the mean of the $n$ attributes of the $j^{th}$ object and Similarity measures between objects that have only
binary attributes are called *similarity coefficients* and have values between 0 and 1. A value of 0 means that the objects are completely dissimilar and a value of 1 means that the objects are completely similar.

Suppose objects \( i \) and \( j \) have \( n \) binary attributes. Then, on comparing \( i \) and \( j \) the following quantities are obtained:

1. \( q_{00} \) the number of attributes where \( i = 0 \) and \( j = 0 \),
2. \( q_{01} \) the number of attributes where \( i = 0 \) and \( j = 1 \),
3. \( q_{10} \) the number of attributes where \( i = 1 \) and \( j = 0 \), and
4. \( q_{11} \) the number of attributes where \( i = 1 \) and \( j = 1 \).

Using the above quantities different similarity coefficients can be obtained:

**Simple Matching Coefficient** or *SMC* [HK06] is one of the most commonly used similarity coefficients and is defined as,

\[
SMC = \frac{\text{Total number of matched attributes}}{\text{Total attributes}} = \frac{q_{00} + q_{11}}{q_{00} + q_{01} + q_{10} + q_{11}}
\]  

(2.14)

*SMC* gives equal weight to both presences and absences.

**Jaccard Coefficient** [HK06] is used for handling objects consisting of asymmetric binary attributes. Jaccard Coefficient (*J*) is defined as follows,

\[
J = \frac{\text{Number of matched attributes}}{\text{Total attributes} - \text{non existence of both attributes}} = \frac{q_{11}}{q_{01} + q_{10} + q_{11}}
\]  

(2.15)

Pearson’s correlation is a powerful similarity measure. However, empirical study has shown that it is not robust with respect to outliers [HMY99], thus potentially yielding false positives which assign a high similarity score to a pair of dissimilar patterns. If two patterns have a common peak or valley at a single feature, the correlation will be dominated by this feature, although the patterns at the remaining features may be completely dissimilar. Another drawback of Pearson’s correlation coefficient is that it assumes an approximate Gaussian distribution of the points and may not be
robust for non-Gaussian distributions [Bic01].

*Jackknife correlation* [JTZ04], helps in overcoming the single outlier problem of Pearson’s correlation. It is defined as,

\[
\text{Jackknife}_{i,j} = \min\{\rho^1_{i,j}, \cdots, \rho^l_{i,j}, \cdots, \rho^n_{i,j}\}
\]

(2.16)

where \(\rho^l_{i,j}\) is the Pearson’s correlation coefficient of data objects \(i\) and \(j\) with the \(l\)th feature deleted. Use of Jackknife correlation avoids the “dominance effect” of single outliers. More general versions of Jackknife correlation that are robust to more than one outlier can similarly be derived. However, generalized Jackknife correlation, which involves the enumeration of different combinations of features to be deleted, is computationally costly and is rarely used.

*Spearman’s rank-order correlation coefficient* is used to address the problem of non-Gaussian distributions w.r.t. Pearson’s correlation, the Spearman’s rank-order correlation coefficient [JTZ04] has been suggested as a similarity measure. The ranking correlation is derived by replacing the data \(x_{in}\) with its rank \(r_{in}\) among all conditions. For example, \(r_{in} = 3\) if \(x_{in}\) is the third highest value among \(x_{if}\), where \(1 \leq f \leq n\). Spearman’s correlation coefficient does not require the assumption of Gaussian distribution and is more robust against outliers than Pearson’s correlation coefficient. However, as a consequence of ranking, a significant amount of information present in the data is lost.

*Cosine Similarity* [TSK09] is useful for finding document similarity. If \(x\) and \(y\) are two document vectors, then \(\cos_{x,y}\) is given by the following equation,

\[
\cos_{x,y} = \frac{x \cdot y}{\|x\| \|y\|}
\]

(2.17)

where \(\cdot\) indicates the vector dot product, \(x \cdot y = \sum_{k=1}^{n} x_k y_k\), and \(\|x\|\) is the length of vector \(x\), and \(\|x\| = \sqrt{\sum_{k=1}^{n} x_k^2} = \sqrt{x \cdot x}\).

*CorHsim*: In [LWN+09], a new similarity measure for gene expression microarray data, *CorHsim*, is presented. It reflects the magnitude and shape information
of gene expression data at the same time and is defined as follows:

\[ \text{CorHsim}_{x,y} = \frac{1}{2n} \sum_{i=1}^{n} \left( \frac{|(x_i - \bar{x})(y_i - \bar{y})|}{\sigma_x \sigma_y} + \frac{1}{1 + |x_i - y_i|} \right) \]  \hspace{1cm} (2.18)

where, \( \sigma_x \) and \( \sigma_y \) are the standard deviations of \( x \) and \( y \) respectively. The disadvantage of \( \text{CorHsim} \) is that it uses the mean value and may sometimes represent the pattern differently.

The similarity/dissimilarity measures discussed above have been applied in various domains. However, not all the measures are applicable throughout all domains. There is a qualitative domain specific dependency among similarity/dissimilarity measures.

2.2 Existing Clustering Approaches

Generally, clustering algorithms are categorized into partitioning methods, hierarchical methods, density-based methods, grid-based methods, model-based methods, graph based methods, cluster ensembles, distributed methods, soft computing methods, and subspace clustering [JMF99].

2.2.1 Partitional

Partitional clustering divides the set of data objects into non-overlapping (disjoint) clusters such that each object is in exactly one cluster. Partitioning methods are divided into two major subcategories [HK06], the centroid based and the medoid based algorithms. The centroid based algorithms represent each cluster by using the gravity center (mean) of the instances while the medoid algorithms represent each cluster by means of the instances closest to the mean.

The k-means algorithm is one of the most well-known centroid algorithm. The k-means method partitions the dataset into \( k \) subsets such that all points in a given subset are closest to the same center. In randomly selects \( k \) of the instances to represent the cluster centers and based on the selected attributes, all remaining instances
are assigned to their nearest cluster center. K-means then computes the new cluster centers by taking the mean of all data points belonging to the same cluster. The process is iterated until some convergence criterion is met (usually till there is no change in the cluster centers). Generally, the k-means algorithm has the following important properties: (i) It is efficient in processing large datasets, (ii) It often terminates at a local optimum, (iii) The clusters have spherical shapes, (iv) It is sensitive to noise. However, the number of clusters have to be provided as an input parameter and choosing the proper initial centroids is the key step of the basic k-means procedure and results are dependent on it.

K-medoid is also a partitional clustering technique that clusters the dataset of $N$ objects into $k$ clusters and is more robust to noise and outliers as compared to k-means. A medoid is the most centrally located object in a given dataset. It can be defined as that object of a cluster, whose average dissimilarity to all the objects in the cluster is minimal[HK06]. One of the most popular k-medoid clustering is the Partitioning around Medoids (PAM) algorithm which begins with an arbitrary set of $k$ objects as medoid points out of $N$ data objects. Each data object in the given dataset is associated to the most similar medoid. Then a non-medoid object, say $o_i$, is selected randomly and the total cost $S_{cost}$ of swapping initial medoid object to $o_i$ is computed. If $S_{cost} < 0$, then swap initial medoid with the new one. The process of selection of medoids and swapping is iterated until there is no change in the medoid.

The k-modes algorithm [Hua98] is a recent partitioning algorithm and uses the simple matching coefficient measure to deal with categorical attributes. For clustering instances described by mixed attributes, a k-prototypes algorithm [Hua98] is also proposed that integrates the k-means and k-modes algorithms and uses a combined dissimilarity measure during clustering. In [Che03], a generalization of conventional k-means clustering algorithm has been presented that is applicable to ellipse-shaped data clusters as well as ball-shaped ones and does not require the exact cluster number apriori.
Hierarchical clustering [HK06, TSK09] provides a nested sequence of partitions, represented graphically with a dendrogram. Each node (cluster) in the tree (except the leaf nodes) is the union of its children (sub-clusters), and the root of the tree is the cluster containing all the objects. Sometimes the leaf nodes consist of a single object and are termed as singleton clusters. Hierarchical methods are divided into two major subcategories: (i) agglomerative method, which forms the clusters in a bottom-up fashion starting with each object in a separate cluster and merging them until all data instances belong to the same cluster and (ii) divisive method, which splits up the dataset into smaller clusters in a top-down fashion until each cluster contains only one instance. Both divisive algorithms and agglomerative algorithms can be represented by dendrograms and are known for their quick termination. Other merits include: (a) they do not require the number of clusters to be known in advance, (b) they compute a complete hierarchy of clusters, (c) good result visualizations are integrated into the methods, and (d) a flat partition can be derived later by cutting through the dendrogram. However, both methods suffer from their inability to perform adjustments once the splitting or merging decision is made. Hierarchical clustering techniques use various criteria to decide locally at each step which clusters should be merged (or split for divisive approaches). To merge or split clusters, the distance between individual objects has been generalized to the distance between subsets. Such derived proximity measure is called a linkage metric. Major inter-cluster linkage includes: single-link, complete-link and average-link [JMF99]. The single-link similarity between two clusters is the similarity between the two most similar instances, one of which appears in each cluster. Single link is good at handling non-elliptical shapes, but is sensitive to noise and outliers. The complete-link similarity is the similarity between the two most similar instances, one of which appears in each cluster. Complete link is less susceptible to noise and outliers, but can break large clusters, and has trouble with convex shapes. The average-link similarity is a compromise between the two. There are various hierarchical clustering algorithms, some of them are: Balanced Iterative Reducing and Clustering using Hierarchies BIRCH [ZRL96], Clustering Using REpresentatives CURE [GRS98] and CHAMELEON [KHK99].
BIRCH [ZRL96] creates a height-balanced tree of nodes that summarize data by accumulating its zero, first, and second moments (CF statistics). It uses a hierarchical data structure called CF-tree for partitioning the incoming data objects in an incremental and dynamic way. CF-tree is a height-balanced tree, which stores the clustering features and it is based on two parameters: branching factor \(B_f\) and threshold \(Th\), which refer to the diameter of a cluster (the diameter (or radius) of each cluster must be less than \(Th\)). A CF tree is built as the data is scanned. While inserting each data object, the CF tree is traversed, starting from the root and choosing the closest node at each level. When the closest leaf cluster for the current data object is finally identified, a test is performed to see if adding the data object to the candidate cluster will result in a new cluster with a diameter greater than the given threshold, \(Th\). If it fits the leaf well and if the leaf is not overcrowded, CF statistics are incremented for all nodes from the leaf to the root. Otherwise a new CF is constructed. Since the maximum number of children per node \((B_f)\) is limited, one or several splits can happen. When the tree reaches the assigned memory size, it is rebuilt and \(Th\) is updated to a coarser one. The outliers are sent to disk, and refitted gradually during tree rebuilds. BIRCH can typically find a good clustering with a single scan of the data and improve the quality further with a few additional scans. It can also handle noise effectively. Moreover, because BIRCH is reasonably fast \((O(N))\), it can be used as a more intelligent alternative to data sampling in order to improve the scalability of other clustering algorithms. However, it may not work well when clusters are not spherical because it uses the concept of radius or diameter to control the boundary of a cluster. In addition, it is order-sensitive as it may generate different clusters for different orders of the same input data. Bubble and Bubble-FM [GRG+98] clustering algorithms are extensions of BIRCH to handle categorical data.

In CURE [GRS98], multiple well-scattered objects (representative points) are chosen to represent a cluster. These points usually capture the geometry and shape of the cluster. The first representative point is chosen to be the point farthest from the cluster center, while the remaining points are chosen so that they are farthest from all previously chosen points. This ensures that the representative points are naturally
relatively well distributed. The number of points chosen, is a parameter, but it was found that a value of 10 or more worked well. The similarity between two clusters is measured by the similarity of the closest pair of the representative points (after they are shrunk toward their respective centers) belonging to different clusters. Once the representative points are chosen they are shrunk toward the center by a factor which ranges between 0 and 1. This helps moderate the effect of outliers, which are usually farther away from the center and thus are shrunk more. CURE uses an agglomerative hierarchical scheme to perform the actual clustering. Unlike centroid/medoid based methods, CURE is capable of finding clusters of different shapes and sizes, as it represents each cluster via multiple representative points. Shrinking the representative points towards the center helps CURE in avoiding the problem of noise. However, it cannot be applied directly to large datasets. For this reason, CURE takes a random sample and performs the hierarchical clustering on the sampled data points.

ROCK [GRS99], is a clustering algorithm for categorical data and uses the Jaccard coefficient as a measure of similarity. It uses the concept of links i.e., the number of common neighbors for any two objects. ROCK first draws a random sample from the dataset and then performs clustering of the data with links. Finally the data in the disk is labeled. It accepts as input the sampled set $S$ to be clustered (that are drawn randomly from the original dataset), and the number of desired clusters $k$. ROCK samples the dataset in the same manner as CURE.

CHAMELEON [KHK99] uses a two-phase approach to cluster the data. In the first phase, it uses a graph partitioning algorithm to divide the dataset into a set of individual clusters. It generates a k-nearest neighbor graph that contains links only between a point and its $k$-nearest neighbors. During the second phase, it uses an agglomerative hierarchical clustering algorithm to find the genuine clusters by repeatedly merging these sub-clusters. None of the clusters formed can contain less than a user specific number of instances. Two clusters are merged only if the inter-connectivity and closeness (proximity) between two clusters are high relative to the internal inter-connectivity of the clusters and closeness of items within the clusters. Therefore, it is better than both CURE and ROCK as CURE ignores information
about inter-connectivity of the objects while ROCK ignores information about the closeness of two clusters.

A novel incremental hierarchical clustering algorithm (GRIN) for numerical datasets based on gravity theory in physics is presented in [CH002]. One main factor that makes the GRIN algorithm able to deliver favorite clustering quality is that the optimal parameters settings in the GRIN algorithm are not sensitive to the distribution of the dataset.

For 2D spatial data (for example, GIS database) the algorithm AMOEBA [ECL00] uses Delaunay diagram (the dual of Voronoi diagram) to represent data proximity and has $O(N\log N)$ complexity. The algorithm consists of two steps: (i) The Delaunay diagram is constructed and a connected planar plane-embedded graph is passed to the algorithm to act as the diagram. Clusters are made up of the points in a connected component and the points in the clusters are reported recursively. Every edge is matched against the criteria and passive edges and noise are discarded; active edges and their points form proximity sub-graphs at each level of the hierarchy. (ii) The algorithm calls itself recursively until no new connected components are created when the passive edges and noise are discarded.

The advantages of hierarchical clustering include: (i) Embedded flexibility regarding the level of granularity, (ii) Ease of handling of any forms of similarity or distance, and (iii) Applicability to any attribute types. The disadvantages of hierarchical clustering are: (i) Vagueness of termination criteria, (ii) The fact that most hierarchical algorithms do not revisit once intermediate clusters that have been constructed with the purpose of their improvement.

2.2.3 Density based

Density-based clustering algorithms try to find clusters based on density of data objects in a region. The key idea of density-based clustering is that for each core object of a cluster the neighborhood of a given radius ($\varepsilon$) has to contain at least a minimum number of instances ($MinPts$), where $\varepsilon$ and $MinPts$ are the two input
parameters. One of the most well known density-based clustering algorithms is the DBSCAN [EKSX96]. This algorithm grows regions with sufficiently high density into clusters. DBSCAN separates data objects into three classes as illustrated in Figure 2.1.

- Core points: These points are at the interior of a cluster. A point is an interior point if there are enough points in its neighborhood.

- Border points: A border point is a point that is not a core point, i.e., there are not enough points in its neighborhood, but it falls within the neighborhood of a core point.

- Noise points: A noise point is any point that is not a core point or a border point.

The neighborhood within a radius \(\varepsilon\) of an object, say \(p\), is called the \(\varepsilon\)-neighborhood of \(p\). If the \(\varepsilon\)-neighborhood of \(p\) contains at least \(\text{MinPts}\) number of objects then \(p\) is a core object. DBSCAN's definition of a cluster is based on the notion of density-reachability. The basic concepts of DBSCAN are directly density-reachability, density-reachability and density connectivity. Basically, an object \(q\) is directly density-reachable from an object \(p\) if it is within the \(\varepsilon\)-neighborhood of a core object, \(p\). An object \(q\) is called density-reachable from \(p\) if there is a sequence of objects \(p_1, p_2, \ldots, p_n\) such that \(p_1 = p\) and \(p_n = q\) where each \(p_{i+1}\) is directly density-reachable from \(p_i\). The relation of density-reachability is not symmetric (since \(q\) might lie on the edge of a cluster, having insufficient number of neighbors for \(q\) to be core). Two objects \(p\) and \(q\) are density-connected if there is an object \(o\) such that both \(p\) and \(q\) are density-reachable from \(o\). A cluster, can therefore be defined as a subset of the objects of the database that satisfies two properties: (i) all objects within the cluster are mutually density-connected and (ii) if an object is density-connected to any point of the cluster, it is part of the cluster as well. To find a cluster, DBSCAN starts with an arbitrary object \((p)\) in dataset \((D)\) and retrieves all objects of \(D\) w.r.t. \(\varepsilon\) and \(\text{MinPts}\). DBSCAN has a number of advantages such as detection of arbitrary shaped clusters, noise handling and is hence quite attractive. However, it suffers from huge computational requirements (the time complexity is \(O(N^2)\)). So it can take huge amounts of time with large datasets. One way to
overcome this problem is to build spatial index structure over the dataset like \( R^* \) tree to locate points within \( \varepsilon \) distance from the core points of the clusters. But this solution is suitable only when the dimensionality of the data is low. Also DBSCAN is dependent on the input parameters \( \varepsilon \) and \( \text{MinPts} \) and there is no straight forward way to fit them to the data. Moreover, different parts of data could require different parameters due to variation in density of the parts.

The algorithm OPTICS (Ordering Points To Identify the Clustering Structure) \cite{ABKS99} can detect clusters of variable density by creating an ordering of the dataset that represents its density-based clustering structure. OPTICS considers a minimum radius \((\varepsilon')\) that makes a neighborhood legitimate for the algorithm. It is a versatile basis for interactive cluster analysis and is consistent with DBSCAN, but goes a step further by keeping the same two parameters \( \varepsilon \), \( \text{MinPts} \) and introducing the concept of core-distance \( \varepsilon' \) (distance to \( \text{MinPts} \) nearest neighbor when it does not exceed \( \varepsilon \), or undefined otherwise). OPTICS covers a spectrum of all different \( \varepsilon' \leq \varepsilon \). The constructed ordering can be used automatically or interactively. With each point, OPTICS stores only two additional fields, the so-called core- and reachability-distances. Experimentally, OPTICS exhibits runtime roughly equal to 1.6 of DBSCAN runtime. While OPTICS can detect the different local densities, it is highly sensitive to its three parameters.

An incremental version of DBSCAN (incremental DBSCAN) is presented in \cite{EKS+98}. It has been proven that this incremental algorithm yields the same result as DBSCAN. In addition, another clustering algorithm (GDBSCAN) generalizing the density-based algorithm DBSCAN is presented in \cite{SEXK98}. GDBSCAN can be applied to both numerical and categorical attributes. Furthermore, DBCLASD (Distribution Based Clustering of Large Spatial Datasets) eliminates the need for \( \varepsilon \) and \( \text{MinPts} \) parameters \cite{XEKS98}. DBCLASD incrementally augments an initial cluster by its neighboring points as long as the nearest neighbor distance set of the resulting cluster still fits the expected distance distribution. DBSCLAD defines a cluster as a non-empty arbitrary shape subset in \( D \) that has the expected distribution of distance to the nearest neighbor with a required confidence, and is the maximal connected set.
with this quality. Regarding connectivity, DBCLASD relies on grid-based approach to generate cluster-approximating polygons. The algorithm contains devices for handling real databases with noise and implements incremental unsupervised learning and can also handle spatial data. Two concepts are used here. First, assignments are not final: points can change cluster membership. Second, certain points (noise) are not assigned, but are tried later. Therefore, once incrementally fetched points can be revisited internally. DBCLASD is known to run faster than CLARANS by a factor of 60 on some examples. In comparison with much more efficient DBSCAN, it can be 2-3 times slower. However, DBCLASD requires no user input, while empirical search for appropriate parameter requires several DBSCAN runs. In addition, DBCLASD discovers clusters of different densities.

Another density-based algorithm is the DENCLUE [HHK98]. The basic idea of DENCLUE is to model the overall point density analytically as the sum of influence functions of the data points. The influence function can be seen as a function, which describes the impact of a data point within its neighborhood. Then, by determining the maximum of the overall density function it can identify the clusters present. The algorithm allows a compact mathematical description of arbitrarily shaped clusters.
in high-dimensional datasets and is significantly faster than the other density based clustering algorithms. Moreover, DENCLUE produces good clustering results even when a large amount of noise is present. As in most other approaches, the quality of the resulting clustering depends on an adequate choice of the parameters. In this approach, there are two important parameters, the parameter $a_1$ determines the influence of a point in its neighborhood and $\eta_1$ describes whether a density-attractor is significant. Density-attractors are local maxima of the overall density function. The runtime of DENCLUE scales with $N$ sub-linearly. This is due to the fact that though all the points are fetched, the bulk of analysis (in clustering stage) involves only points in highly populated areas.

In [JPZ03], the Density-based Hierarchical Clustering method (DHC) is presented. It considers a cluster as a high-dimensional dense area, where data objects are attracted to each other. At the core part of the dense area, objects have higher density whereas objects at the peripheral area are relatively sparse. Once the density and attraction of data objects are defined, DHC organizes the cluster structure of the dataset in two-level hierarchical structures: attraction tree (represents relationships of objects in the dense area) and density tree (summarizes the cluster structure of the attraction tree where each node represents a dense area). Density tree is mined (split into sub-dense areas based on some criteria) for the final clusters. DHC is effective for the high-connectivity characteristic of gene expression data because it first captures the core part of the cluster and then divides the borders of the clusters on the basis of the attraction between the data objects. The two-level hierarchical representation of the dataset enables the relationship among the clusters and also organizes the relationship among the data objects within the same cluster. The computational complexity of this step is $O(N^2)$, which makes DHC inefficient. Furthermore, two global parameters used in DHC to control the splitting process of dense areas are also sensitive.

FDC algorithm (Fast Density-Based Clustering) is presented in [ZCK99] for density-based clustering defined by the density-linked relationship. The clustering in this algorithm is defined by an equivalence relationship on the objects in the database.
The complexity of FDC is linear to the size of the database, which is much faster than that of the algorithm DBSCAN. More recently, the algorithm SNN (Shared Nearest Neighbors) [ESK03] blends a density based approach with the idea of ROCK. SNN sparsifies similarity matrix by only keeping k-nearest neighbors, and thus derives the total strength of links for each object.

2.2.4 Grid based

Grid-based clustering algorithms first quantize the clustering space into a finite number of cells (hyper-rectangles) and then perform the required operations on the quantized space. Cells that contain more than certain number of points are treated as dense and the dense cells are connected to form the clusters. Here, we report some of the grid-based clustering algorithms such as STatistical INformation Grid-based method - STING [WYM97], WaveCluster [SC+98], and CLustering In QUEst - CLIQUE [AGGR98].

STING [WYM97] is a grid based multi resolution clustering technique in which the spatial area is divided into rectangular cells in order to form a hierarchical structure. The cells in a high level are composed from the cells in the lower level. Each cell has four (default) children and stores a point count, and attribute-dependent measures: mean, standard deviation, minimum, maximum, and distribution type. Measures are accumulated starting from bottom level cells, and further propagate to higher-level cells (e.g., minimum is equal to a minimum among the children-minimums). It generates a hierarchical structure of the grid cells so as to represent the clustering information at different levels. Therefore, STING constructs data summaries and assembles statistics in a hierarchical tree of nodes that are grid-cells. Although STING generates good clustering results in a short running time, there are two major problems with this algorithm. Firstly, the performance of STING relies on the granularity of the lowest level of the grid structure. Secondly, the resulting clusters are all bounded horizontally or vertically, but never diagonally. This shortcoming might greatly affect the cluster quality.

CLIQUE [AGGR98] uses the concepts of density and grid based methods. CLIQUE
starts by finding all the dense areas in the one-dimensional spaces corresponding to
each attribute. CLIQUE then generates the set of two-dimensional cells that might
possibly be dense, by looking at dense one-dimensional cells, as each two-dimensional
cell must be associated with a pair of dense one-dimensional cells. The dense units are
then connected to form clusters. It uses apriori algorithm (bottom up algorithm) to
find dense units. Generally, CLIQUE generates the possible set of n-dimensional cells
that might possibly be dense by looking at dense \((n-1)\) dimensional cells. CLIQUE
is able to find clusters in all subspaces of the original data space and present a min-
imal description of each cluster in the form of a DNF expression. Steps involved
in CLIQUE is i) identification of subspaces (dense Units) that contain cluster ii)
merging of dense units to form cluster and iii) Generation of minimal description for
the clusters. CLIQUE produces identical results irrespective of the order in which
the input records are presented. In addition, it generates cluster descriptions in the
form of DNF expressions [AGGR98] for ease of comprehension. Moreover, empirical
evaluation shows that CLIQUE scales linearly with the number of instances, and has
good scalability as the number of attributes is increased.

The algorithm WaveCluster [SC+98] works with numerical attributes and has an
advanced multi-resolution. The main idea is to transform the original feature by ap-
plying wavelet transform and then find the dense regions in the new space. A wavelet
transform is a signal processing technique that decomposes a signal into different fre-
quency sub bands. The first step of the WaveCluster algorithm is to quantize the
feature space. In the second step, discrete wavelet transform is applied on the quan-
tized feature space and hence new units are generated. WaveCluster connects the
components in 2 set of units and they are considered as cluster. Corresponding to
each resolution of wavelet transform there would be set of clusters \(k\), where usually
at the coarser resolutions number of cluster is less. In the next step, WaveCluster
labels the units in the feature space that are included in the cluster. WaveCluster
gives high quality of clusters, can work well in relatively high dimensional spatial
data and can successfully handle outliers. The algorithm's complexity is \(O(N)\) for
low dimensions, but with the increase in the number of dimensions it grows exp-
ponentially. Unlike other clustering methods, WaveCluster [SC+98] does not require
users to give the number of clusters. It is a very powerful method and automatically removes outliers, however, it is not efficient in high dimensional space.

2.2.5 Model based

AutoClass [CS96] uses the Bayesian approach, starting from a random initialization of the parameters, incrementally adjusts them in an attempt to find their maximum likelihood estimates. Another model based method is the SOM net [Koh95] which is based on a single layered neural network. The data objects are organized with a simple 2-D grid structure. Each neuron of the neural network is associated with a reference vector, and each data point is mapped to the neuron with the closest reference vector. In the process of running the algorithm, each data object acts as a training sample which directs the movement of the reference vectors towards the denser areas of the input vector space, so that those reference vectors are trained to fit the distributions of the input dataset. When the training is complete, clusters are identified by mapping all data points to the output neurons. An important property of the SOM is that it is very robust. The outlier can be easily detected from the map, since its distance in the input space from other units is large. SOM can deal with missing data values, too. It generates intuitive cluster patterns of a high dimensional dataset. However, it suffers from some disadvantages such as the number of clusters and the grid structure of the neuron map need to be given as input. It is also sensitive to the input parameter.

Though the model based approach can be considered as more relevant to the data mining problem, most of the existing methods under the approach suffer from the following disadvantages: (i) they try to fit a mathematical model to the data which may not be effective to all domains, (ii) the number of clusters and the grid structure need to be given as input, (iii) they are sensitive to the input parameters and (iv) the algorithms are not cost effective.
2.2.6 Graph Based

Graph theoretical clustering techniques represent the data in terms of a graph, thus converting the problem of clustering a dataset into such graph theoretical problems as finding minimum cut or maximum cliques in a proximity graph [BDSY99]. AUTOCLUST [LECO00] is a graph based algorithm that automatically extracts boundaries based on Voronoi modeling and Delaunay Diagrams. Parameters required are not specified by users but are revealed from the proximity structures of the Voronoi modeling, and AUTOCLUST calculates them from the Delaunay Diagram. This removes human-generated bias and also reduces the exploration time. The advantages are: (i) it is effective in the detection of clusters of different densities, and (ii) it identifies and removes multiple bridges linking clusters and has a complexity of $O(N\log N)$.

CLuster Identification via Connectivity Kernels (CLICK) [SS00] tries to identify clusters as a highly connected component in a proximity graph based on a probabilistic assumption and can detect intersecting clusters. Cluster Affinity Search Technique (CAST) [BDSY99] is based on the concept of a corrupt clique graph data model. CAST assumes that the true clusters of the data points are obtained by a disjoint union of complete sub-graphs where each clique represents a cluster; where a cluster is a set of high affinity elements subject to a threshold. CAST discovers clusters one at a time. Both CAST and CLIQUE are popular for detecting clusters over gene expression data. The graph theoretic approach can be considered to be more relevant to gene expression data mining as they are capable of discovering intersected and embedded clusters. However, it sometimes generates non-realistic cluster patterns.

There are many applications that require the clustering of large amounts of high dimensional data. However, most automated clustering techniques do not work effectively and/or efficiently on high dimensional data, i.e. they often miss clusters with certain unexpected characteristics. The reasons for this are: (i) it is difficult to estimate the necessary parameters for tuning the clustering algorithms to the specific application’s characteristics, (ii) it is hard to verify and interpret the resulting high dimensional clusters and (iii) often the concept of clusters inspired from low dimensional cases cannot be extended to high dimensional cases. A solution to these problems may be obtained by integrating all the requirements into a single algorithm.
and to try to build a combination of clustering algorithms (ensembles of clustering algorithms)

2.2.7 Ensembles of Clustering Algorithms

In the combination of techniques in a group or ensemble, the outputs provided by different techniques are combined by one of several strategies in order to provide a consensus output value [HCFdC09]. The main goal is to use the best features of each individual technique and improve the overall performance in terms of accuracy or precision. The theoretical foundation of combining multiple clustering algorithms is still in its early stages. According to [HK07, HKK05], clustering ensembles are formed by the combination of a set of partitions previously produced by several runs of a single algorithm or by a set of algorithms. Since, there is no label associated with each object, some form of sophisticated strategies are needed in order to combine partitions found by different algorithms or different runs of the same algorithm in a consensus partition. Combining multiple clustering algorithms is a more challenging problem than combining multiple classifiers. Clustering combination a difficult task because various clustering algorithms produce very different results due to different clustering criteria, combining these clustering results directly may not generate a good meaningful result. According to [SG03], cluster ensembles can be formed in a number of different ways, such as (i) the use of a number of different clustering techniques (either deliberately or arbitrarily selected), (ii) the use of a single technique many times with different initial conditions and/or (iii) the use of different partial subsets of features or patterns. In [FJ02], a split-and-merge strategy is followed. In the first step, k-means algorithm is used to generate small, compact clusters. An ensemble of clustering algorithms is produced by random initializations of cluster centroids. Data partitions present in these clustering are mapped into a new similarity matrix between patterns, based on a voting mechanism. This matrix, is independent of data sparseness, is then used to extract the natural clusters using the single link algorithm. In [AK03], multiple clustering algorithms were combined based on a Weighted Shared nearest neighbors Graph method. In [YAL+06] multiple crossover repetitions were used to combine partitions created by different clustering algorithms. Each pair selected for a crossover operation should present a high overlap
in the clusters. The initial population comprises of all clusters created by the clustering algorithms used in the ensemble. This method, named heterogeneous clustering ensemble (HCE), differ from other ensemble approaches by taking characteristics from the individual algorithms and the dataset into account during the ensemble procedure. This method was compared with individual clustering algorithms using a gene expression dataset.

Due to the increasing size of current databases, constructing efficient distributed clustering algorithms has attracted considerable attention.

2.2.8 Distributed Clustering

Distributed Clustering assumes that the objects to be clustered reside on different sites. Instead of transmitting all objects to a central site (also known as server) where we can apply standard clustering algorithms to analyze the data (also known as sequential clustering), the data are clustered independently on different local sites. The central site updates the global clustering based on the local models, i.e. the representative clustering transmitted from the local sites. Generally, as far as distributed clustering is concerned, there are different scenarios: (i) Feature-Distributed Clustering (FDC), combines a set of clusterings obtained from clustering algorithm having partial view of the data features, (ii) Object-Distributed Clustering (ODC), combines clusterings obtained from clustering algorithm that have access to the whole set of data features and to a limited number of objects, and (iii) Feature/Object-Distributed Clustering (FODC), consists in combining clusterings obtained from clustering algorithm having access to limited number of objects and/or features of the data. Various distributed clustering techniques have been proposed such as a parallel version of the k-means algorithm was proposed in [DM99], a parallel version of DBSCAN, called PDBSCAN was presented in [XJK99] that uses a shared-nothing architecture with multiple computers interconnected through a network. PDBSCAN offers nearly linear speedup and has excellent scale-up and size-up behavior. The Density Based Distributed Clustering (DBDC) algorithm [JKP03] can be used in the case when the data to be clustered is distributed and infeasible to centralize. A detailed survey of distributed clustering is reported in Section 5.2.
2.2.9 Soft Computing

Traditional clustering approaches generate disjoint groups or clusters. Fuzzy clustering on the other hand associates each pattern with every cluster using a membership function with larger membership values indicating higher confidence in the assignment of the pattern to the cluster. One widely used fuzzy clustering algorithm is the Fuzzy C-Means (FCM) algorithm [Bez81a], which is based on k-means. FCM attempts to find the most characteristic point in each cluster, which can be considered as the cluster center and, then, the degree of membership for each object in the clusters are computed. The work in [AZM06] attempts to segment satellite image based on FCM algorithm and to detect different road classes on it. Some variants of fuzzy clustering for satellite image domain are presented in [ACN09, AN09, GyFSr09]. In [VB09], a density based clustering method called rough-DBSCAN is presented. It is a modification of the well known density based clustering method DBSCAN [EKX96] and aims at achieving similar result as DBSCAN but in much smaller time requirement (O(N)).

Neural Networks-based clustering approaches have also gained popularity in recent years. Examples are SOFM (Self Organizing Feature Map) [Koh95, AN09] and ART (Adaptive Resonance Theory) [THHK02]. SOFM attempts to visualize a high dimensional input pattern with prototype vectors in a two-dimensional lattice structure, where each node in the lattice structure is a neuron, which are connected to each other via adaptable weights. During the training process, the neighboring input patterns are projected into the lattice corresponding to adjacent neurons. The advantages of SOFM are: (i) It enjoys the benefits of input space density approximation and, (ii) it is input order independent. The disadvantages are (i) like k-means, SOFM needs to predefine the size of the lattice, (the number of clusters) and, (ii) it may suffer from input space density misrepresentation. ART [THHK02] is a large family of neural network architecture and is capable of learning any input pattern in a fast, stable and self-organizing way.

Genetic Algorithms (GA) [Gol89] are also used in cluster analysis. GA clustering is basically a randomized search and optimization technique based on the principles of
evolution and natural genetics. Several GA-based clustering algorithms are found in the literature [ACN09, LLF+04b, KM99, LLF+04a, BP01, MB03a, BMM07a]. GAs have also been used to cluster satellite images such as the real-coded variable string length genetic fuzzy clustering in [MB03a] and multi-objective optimization algorithm in [BMM07a]. GGA (Genetically Guided Algorithm) is a genetic algorithm for fuzzy and hard k-means [HOB99]. Evolutionary techniques rely on certain parameters to empirically fit data and have high computational costs that limit their application in data mining. However, usage of combined strategies (e.g., generation of initial guess for k-means) has been attempted [BM93].

Other soft clustering algorithms have been developed and most of them are based on the Expectation-Maximization (EM) algorithm [DLR77]. They assume an underlying probability model with parameters that describe the probability that an instance belongs to a certain cluster. EM algorithm starts with initial guesses for the mixture model parameters. These values are then used to calculate the cluster probabilities for each object which in turn are used to re-estimate the parameters, and the process is repeated. A drawback of such algorithms is that they tend to be computationally expensive. Another problem found in this approach is called overfitting. This problem might be caused due to two reasons. On one hand, a large number of clusters may be specified and on the other, the distributions of probabilities have too many parameters. In this context, one possible solution is to adopt a fully Bayesian approach, in which every parameter has a prior probability distribution.

In case of fuzzy clustering, the problem of specifying the number of clusters exists. The basic advantage of ART is that it is fast, exhibits stable learning and pattern detection. The disadvantage is its inefficiency in handling noise and higher dimensional representation for clusters. GA-based clustering has also been used extensively recently, however, solutions are not always free from the local optima problem.

2.2.10 Subspace Clustering

Subspace clustering was initially proposed by Agrawal et al. [AGGR98], to evaluate features on only a subset of the data, based on a measure referred to as a “measure
of locality" representing a cluster. In this subsection, subspace clustering algorithms are discussed in four broad categories.

a. Bottom-Up SubspaceSearch Methods take the advantage of downward closure of the property of density to reduce the search space. It determines locality by creating bins for each dimension which finally form multi-dimensional grid, achieved by two approaches: (i) static grid-sized approach, e.g., CLIQUE [AGGR98] and ENCLUS[CFZ99], two popular algorithms of this category, and (ii) data driven strategies adopted to determine the cut-points, e.g., MAFIA, CBF, CL Tree and DOC. CLIQUE [AGGR98] can find clusters within subspaces using a grid-density based clustering approach and is capable of identifying arbitrary shaped clusters in any number of dimensions without specifying the number of clusters. The clusters may be found in the same space or in overlapping and disjoint subspaces. CLIQUE scales well with the number of instances and dimensions in the dataset. This method is not without its disadvantages, which are both grid size and the density threshold are input parameters which affect the quality of the clustering results, the small but important clusters can sometimes be eliminated during the pruning stage. ENCLUS [CFZ99] is a bottom-up clustering method which defines clusters based on entropy and can locate overlapping clusters of various shapes in subspaces of different sizes. However, its scalability is poor with respect to the subspace dimensions. Merging of Adaptive Finite Intervals Algorithm (MAFIA) [GGN+99] is a variant of CLIQUE and uses an adaptive, grid-based approach with parallelism, to improve scalability. The advantages of MAFIA are that it can locate clusters of various sizes and shapes, its performance is faster than CLIQUE due to the adoption of parallel approach and its scale-up is linear. However, the running time grows exponentially as the number of dimensions increase. Cell-Based Clustering (CBF) [CJ02] is also a bottom-up algorithm but unlike CLIQUE and MAFIA, it uses an efficient algorithm for creation of partitions optimally, to avoid exponential growth of bins with the increase in the number of dimensions. CBF locates clusters of various sizes and shapes, scales linearly with respect to the number of records in a dataset and its performance is better since the bins are stored in an index structure. But, it is sensitive
to the threshold which determines the bin frequency of a dimension and the threshold which determines the number of data points in a bin. Density-based Optimal projective Clustering (DOC) [PJAM02] is basically a hybridization of bottom-up and top-down approaches. It introduces the notion of an optimal projective cluster. The advantage is that the running time grows linearly with the number of instances, whereas the disadvantages are it is sensitive to the input parameters, is able to identify mostly hyper-rectangular shaped clusters, and the running time grows exponentially with the increase in the number of dimensions in the dataset.

b. Top-Down SubspaceSearch Methods start with an initial approximation of clusters over an equally weighted full feature space. Next, it follows an iterative procedure to update the weights and accordingly reforms the clusters. It is an expensive clustering algorithm over the full feature space. However, the use of sampling technique can improve the performance. The number of clusters and the size of the subspace are the most critical factors in this approach. PROCLUS [AWY+99] is a sampling biased top-down subspace clustering algorithm which randomly selects a set of k-medoids from a sample and iteratively improves the choice of medoids to form better clusters. The disadvantages of this method are it is biased towards hyper-spherical shaped clusters, clustering quality depend upon the size of the sample chosen, and it is sensitive to the input parameters. ORCLUS [AY00] attempts to form clusters iteratively by assigning the points to the nearest cluster representation. It computes the dissimilarity between a pair of points as a set of orthonormal vectors over a subspace. It is a fast and a scalable method. However, it requires the size of the subspace dimensionality and the number of clusters apriori and may sometimes miss some small clusters. The algorithm δ-Clusters [YWWY02] starts with an initial seed and attempts to improve the overall quality of the cluster iteratively by swapping dimensions with instances. The advantage is that the use of coherence as a similarity measure makes it more relevant for microarray data analysis and its disadvantages are that (i) it is dependent on two input parameters which are number and size of the cluster (ii) the running time is dependent upon the cluster size. COSA [FM04] starts with an equally weighted
dimension for each instance and then it examines the $k$-nearest neighbors (knn) of an instance. Based on knn, it calculates the respective dimension weights for each instance and assigns higher weighted dimensions to those instances which have lesser dispersion within the neighborhood. This process is then repeated with the new instances till the weights stabilize. The advantage of this method is that the number of dimensions in clusters need not be specified and the dimension weights are calculated.

c. **Biclustering Algorithms:** A bicluster [CC00] is an $I \times J$ sub-matrix that exhibits some coherent tendency where $I$ and $J$ are set of genes (rows) and conditions (columns), respectively, and $|I| \leq |G|$ and $|J| \leq |T|$. The volume of a bicluster $(I, J)$ is defined as the number of elements $e_{ij}$ such that $i \in I$ and $j \in J$. The quality of a bicluster is assessed based on the mean squared residue, which is the variance of the set of all elements in the bicluster, plus the mean row variance and the mean column variance. The lower the mean squared residue, stronger is the coherence exhibited by the cluster and better is the quality of the bicluster. Cheng and Church [CC00] were the pioneers in applying biclustering to gene expression data. To obtain the larger bi-clusters with minimum mean squared residue, the authors introduced the node addition method which simultaneously adds rows/columns as deletions took place. However, it is not capable of identifying overlapping/embedded clusters because the elements of the already identified bicluster are masked by random noise. **Flexible Overlapped biClustering (FLOC)** [YWWY03] addresses the limitation of Cheng and Church [CC00] and accelerates the biclustering process, FLOC uses a probabilistic algorithm which can discover a set of k-possible overlapping biclusters simultaneously. **Order-preserving submatrices (OPSM)** [BdCKY02] is another probabilistic model which attempts to address the idea of large OPSMs with maximum statistical significance. Tanay et al. in [TSS02] introduced Statistical-Algorithmic Method for Bicluster Analysis (SAMBA), a bi-clustering algorithm that performs simultaneous bicluster identification by using exhaustive enumeration. An important advantage of SAMBA is that it is capable of analyzing large datasets in lesser time. The **Coupled Two-Way Clustering (CTWC)** [GLD00] tries to identify couples of small subsets of fea-
tures and objects. The Interrelated Two-Way Clustering (ITWC) [TZRZ01] is an iterative biclustering algorithm based on a combination of the results obtained by clustering performed on each of the two dimensions of the data matrix separately.

d. *TriClustering* Algorithms tries to find coherent clusters along gene-sample-time (temporal) or gene-sample-region (spatial) dimensions, known as triclusters, which may be arbitrarily positioned and overlapped [ZZ05]. TriClustering algorithms are used for mining such coherent clusters in three-dimensional gene expression datasets. TriCluster relies on a graph-based approach to mine various types of clusters, including clusters having constant or similar values along each dimension with scaling and shifting expression patterns, based on different parameter values.

### 2.2.11 A General Comparison among Different Approaches

We have discussed some of the unsupervised clustering methods present in the literature. Partitioning algorithms typically represent clusters by a prototype and an iterative control strategy is used to optimize the whole clustering such as the average or squared distances of instances to its cluster centers (prototypes) are minimized. Partitional clustering algorithms are effective for determining clusters of convex shape, similar size and density, and if the number of clusters can be reasonably estimated. However, determining the appropriate number of clusters is very difficult. Hierarchical clustering algorithms decompose the dataset into several levels of partitioning and are represented by a tree structure which splits the dataset recursively into smaller subsets. Although hierarchical clustering algorithms can be very effective in knowledge discovery, the cost of creating the tree is very expensive for large datasets. In density-based approaches clusters are regarded as regions in the data space where the objects are dense, and they are separated by regions of low density (noise). These regions may have an arbitrary shape and the objects inside a region may be arbitrarily distributed. Generally, grid-based clustering algorithms first separate the clustering space into a finite number of cells (hyper-rectangles) and then perform the required operations on the quantized space. Cells that contain more than certain number of
points are treated as dense and the dense cells are connected to form the clusters. A solution for better results could be instead of integrating all the requirements into a single algorithm, to try to build a combination of clustering algorithms. In addition, the impact of various soft computing techniques is also considerable for identifying clusters that are not crisp. The performance and quality of distributed and parallel clustering techniques have helped in managing and processing massive data. For high dimensional clustering, subspace clustering algorithms have given quite good results. A performance comparison of various clustering algorithms is given in Table 2.1.

2.2.12 Handling Outliers

Data usually have an associated amount of noise, which can be viewed as outliers. Alternately, outliers can be viewed as legitimate records having abnormal behavior. The algorithm BIRCH [ZRL96] revisits outliers during the major CF tree rebuilds, but in general handles them separately. Some algorithms have specific features for outliers handling. The algorithm CURE [GRS98] uses shrinkage of cluster representatives to suppress the effects of outliers. K-medoids methods are generally more robust than k-means methods with respect to outliers. The algorithm DBSCAN [EKSX96] uses concepts of internal (core), boundary (reachable), and outliers (non-reachable) points. CLIQUE [AGGR98] eliminates subspaces with low coverage. WaveCluster [SC+98] handles outliers very well through its filtering process. The algorithm ORCLUS [AYOO] produces a partition plus a set of outliers. A point is said to be an outlier if its $\varepsilon$-neighborhood contains less than $MinPts$-fraction of a whole dataset $D$ [KNT00]. In essence, different subsets of data have different densities and may be governed by different distributions. A point close to a tight cluster can be a more probable outlier than a point that is further away from a more dispersed cluster. The concept of local outlier factor (LOF) that specifies a degree of outlier-ness comes to rescue [BKNS00]. The definition is based on the distance to the k-nearest neighbor. Knorr et al. [KNZ01] addressed a related problem of how to eliminate outliers in order to compute an appropriate covariance matrix that describes a given locality. Outlier detection techniques can be divided into the following categories given below.
• **Distribution-based methods** handle one dimensional data and assume a statistical distribution such as Gaussian and try to fit the data to the model by estimating the parameters such as mean and variance from the data [BL94]. They vary in terms of type of distribution, number of outliers to be identified and type of outliers. Then they employ a test based on the distribution property to identify outliers w.r.t. this distribution. In reality, prior knowledge about the distribution of the dataset is not always available.

• **Depth-based approaches** [RR96, JKN98] employ computational geometry to compute different layers of convex hulls and declare those objects in the outer layer as outliers. However, they suffer from the dimensionality curse and cannot cope with large dimensions.

• **Distance-based approaches** distinguish points which are likely to be outliers from others based on the number of points in their neighborhood and are suitable for finding outliers in large datasets. Corresponding to clustering algorithms that find convex clusters [KR90, NH02], one well known technique is the $DB(p,d)$-outlier [KN98], where a point in a dataset $D$ is an outlier if at least $p$ fraction of points in $D$ lie greater than distance $d$ from it. A special case of $DB(p,d)$-outlier is proposed in [RRS00], where the distance to the k-th nearest neighbor is used to rank the outlyingness. But, it cannot handle data with different local densities and hence can only find global outliers. Besides, the users parameters, such as $p, d, k$, are hard to determine.

• **Density-based approaches** focus on the local density comparison only with the immediate neighbors. They come in two classes, subspace and full space. Sometimes, an object could reside in a low density region only in a subspace, which is obtained by projecting the original full space onto one of its subsets. In [AGGR98], all possible subspaces were searched where there are regions with much lower density than the rest of the subspace. All points in those low density regions are declared as outliers. In [BKNS00], the authors introduced the notion of LOF, which measures the degree of outlyingness, based on the difference in the local density of a point and its k-nearest neighbors. $DB(p,d)$-outlier cannot detect local outliers w.r.t. a neighboring dense cluster in presence of
another very sparse cluster because although the local density of the outlier can be lower than those inside the neighboring high density cluster, it may be comparable to those inside the sparse (low density) cluster, therefore, a large portion of points in the sparse cluster will also be classified as outliers. LOF solves this problem by comparing local density of the outlier only with those of its neighboring objects. The weakness of LOF is that it cannot detect outliers whose local density is higher, not lower, than those inside the neighboring pattern. Such a pattern may consist of a set of regularly spaced points that have lower densities than their neighboring outliers.

2.3 Discussion

This chapter presents various proximity measures, clustering and outlier detection techniques. Clustering algorithms are dependent on the proximity measure chosen. Moreover, there exists no particular measure which can handle all the issues or domains. From the discussion above, we conclude that various clustering algorithms require different types of input parameters and clustering results are highly dependent on the values of the parameters. Clustering algorithms that can handle massive data, identify clusters even in high dimensional and noisy data are in great demand. Again, identification of variable density clusters is an important research area which has gained focus due to its huge potential. Clustering algorithms that do not require the number of clusters beforehand, insensitive to the proximity measure, and robust to noise are of utmost importance.

In this thesis, we present several clustering techniques for application over 2D spatial data, satellite and gene expression data. The advantages of our techniques are: (i) Independence of the number of clusters, (ii) detection of variable density clusters, (iii) identifications of sub-clusters, (iv) capability in detecting clusters in large-scale data and (v) handling outliers and noise. The next chapter presents our first clustering technique capable of identifying variable density clusters in 2D spatial data.
Table 2.1: Comparison of various clustering algorithms

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>No. of Parameters</th>
<th>Optimized For</th>
<th>Structure</th>
<th>Multi-Density Clusters</th>
<th>Embedded Clusters</th>
<th>Complexity Handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-means</td>
<td>No. of clusters</td>
<td>Separated Clusters</td>
<td>Spherical</td>
<td>No</td>
<td>No</td>
<td>$O(l^2kN)$ No</td>
</tr>
<tr>
<td>k-modes</td>
<td>No. of clusters</td>
<td>Separated Clusters, large datasets</td>
<td>spherical</td>
<td>No</td>
<td>No</td>
<td>$O(l^2kN)$ No</td>
</tr>
<tr>
<td>FCM</td>
<td>No. of clusters</td>
<td>Separated Clusters</td>
<td>Non-convex shapes</td>
<td>No</td>
<td>No</td>
<td>$O(N)$ No</td>
</tr>
<tr>
<td>PAM</td>
<td>No. of clusters</td>
<td>Separated Clusters, small datasets</td>
<td>spherical</td>
<td>No</td>
<td>No</td>
<td>$O(l^2k(N - k)^2)$ No</td>
</tr>
<tr>
<td>CLARA</td>
<td>No. of clusters</td>
<td>relatively large datasets</td>
<td>spherical</td>
<td>No</td>
<td>No</td>
<td>$O(k \times S + k(N - k))$ No</td>
</tr>
<tr>
<td>CLARANS</td>
<td>No. of clusters, Max no. of neighbors</td>
<td>better than PAM, CLARA</td>
<td>spherical</td>
<td>No</td>
<td>No</td>
<td>$O(kN^2)$ No</td>
</tr>
<tr>
<td>BIRCH</td>
<td>Branching factor, Diameter, Threshold</td>
<td>Large data</td>
<td>Spherical</td>
<td>No</td>
<td>No</td>
<td>$O(N)$ Yes</td>
</tr>
<tr>
<td>CURE</td>
<td>No. of clusters, No. of representatives</td>
<td>Any shape large data</td>
<td>Arbitrary</td>
<td>No</td>
<td>No</td>
<td>$O(N^2\log N)$ Yes</td>
</tr>
<tr>
<td>ROCK</td>
<td>No. of clusters</td>
<td>Small noisy data</td>
<td>Arbitrary</td>
<td>No</td>
<td>No</td>
<td>$O(N^2 + N_{mm}m_a + N^2\log N)$ Yes</td>
</tr>
<tr>
<td>CHAMELEON</td>
<td>3 (k-nearest neighbor, MINSIZE, $\alpha^c$)</td>
<td>Small datasets</td>
<td>Arbitrary</td>
<td>Yes</td>
<td>No</td>
<td>$O(N^2)$ Yes</td>
</tr>
<tr>
<td>Algorithms</td>
<td>No. of Parameters</td>
<td>Optimized For</td>
<td>Structure</td>
<td>Multi-Density Clusters</td>
<td>Embedded Clusters</td>
<td>Complexity</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td>---------------</td>
<td>-----------------</td>
<td>------------------------</td>
<td>-------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>2 ((\text{MinPts}, \varepsilon))</td>
<td>Large datasets</td>
<td>Arbitrary</td>
<td>No</td>
<td>No</td>
<td>(O(N \log N)) using (R^*) tree</td>
</tr>
<tr>
<td>OPTICS</td>
<td>3 ((\text{MinPts}, \varepsilon, \varepsilon'))</td>
<td>Large datasets</td>
<td>Arbitrary</td>
<td>Yes</td>
<td>Yes</td>
<td>(O(N \log N)) using (R^*) tree</td>
</tr>
<tr>
<td>DENCLUE</td>
<td>2 ((\text{MinPts}, \varepsilon))</td>
<td>Large datasets</td>
<td>Arbitrary</td>
<td>No</td>
<td>No</td>
<td>(O(N \log N)) using (R^*) tree</td>
</tr>
<tr>
<td>WaveCluster</td>
<td>No. of cells for each dimension, No. of applications of transform</td>
<td>Any shape, Large data</td>
<td>Any</td>
<td>Yes</td>
<td>No</td>
<td>(O(N))</td>
</tr>
<tr>
<td>STING</td>
<td>No. of cells in lowest level, No. of objects in cell</td>
<td>Large spatial datasets</td>
<td>Vertical and horizontal boundary</td>
<td>No</td>
<td>No</td>
<td>(O(N))</td>
</tr>
<tr>
<td>CLIQUE</td>
<td>Size of the grid, minimum no. of points in each grid cell</td>
<td>High dimensional, large datasets</td>
<td>Arbitrary</td>
<td>No</td>
<td>No</td>
<td>(O(N))</td>
</tr>
<tr>
<td>MAFIA</td>
<td>Size of the grid, minimum no. of points in each grid cell</td>
<td>High dimensional, large datasets</td>
<td>Arbitrary</td>
<td>No</td>
<td>No</td>
<td>(O(c^d\varepsilon))</td>
</tr>
<tr>
<td>AUTOCLUST</td>
<td>NIL</td>
<td>Massive data</td>
<td>Arbitrary</td>
<td>No</td>
<td>No</td>
<td>(O(N \log N))</td>
</tr>
</tbody>
</table>