CLUSTERING

Clustering Analysis: Overview, K-Means, Agglomerative Hierarchical Clustering, DBSCAN, Cluster Evaluation, Density-Based Clustering, Graph- Based Clustering, Scalable Clustering Algorithms

8.1Overview:

Cluster analysis groups data objects based only on information found in the data that describes the objects and their relationship.

Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups. The greater the similarity within a group and the grater the difference between groups, the better or more distinct Clustering.

There are 20 points and divide them into 3 different ways of clustering.

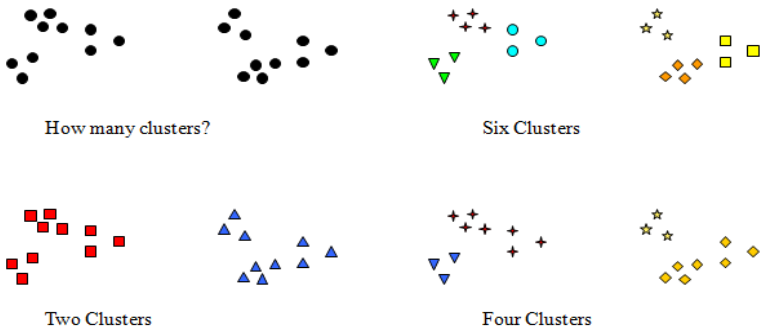


Figure 8.1. Different ways of clustering the same set of points.

Cluster analysis is related to other techniques that are used to divide data object into groups.

supervised classification; i.e., new, unlabeled objects are assigned a class label using a model developed from objects with known class labels. For this reason, cluster analysis is sometimes referred to as unsupervised classification. When the term classification is used without any qualification within data mining, it typically refers to supervised classification.

Simple segmentation

– Dividing students into different registration groups alphabetically, by last name.

8.1.1 Different Types of Clusterings

An entire collection of clusters is commonly referred to as a clustering, .

we distinguish various types of clusterings: hierarchical (nested) versus partitional (unnested), exclusive versus overlapping versus fuzzy, and complete versus partial.

Hierarchical versus Partitional :

A partitional clustering is simply a division of the set of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset.

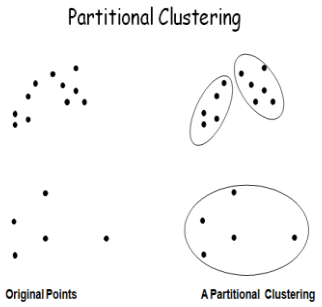
Taken individually, each collection of clusters in Figures 8.1 is a partitional clustering.

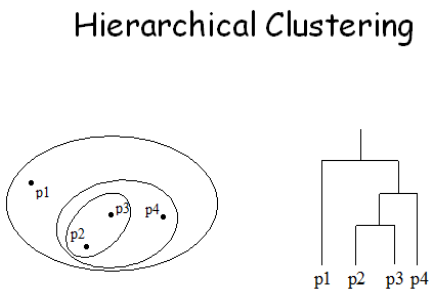
If we permit clusters to have subclusters, then we obtain a hierarchical clustering, which is a set of nested clusters that are organized as a tree.

Each node (cluster) in the tree (except for the leaf nodes) is the union of its children (subclusters), and the root of the tree is the cluster containing all the objects.

Often, but not always, the leaves of the tree are singleton clusters of individual data objects.

Finally, note that a hierarchical clustering can be viewed as a sequence of partitional ciusterings and a partitional clustering can be obtained by taking any member of that sequence; i.e., by cutting the hierarchical tree at a particular level.





Exclusive versus Overlapping versus Fuzzy :

The clusterings shown in Figure 8.1 are all exclusive, as they assign each object to a single cluster.

There are many situations in which a point could reasonably be placed in more than one cluster, and these situations are better addressed by non-exclusive clustering(overlapping). In the most general sense, an overlapping or non-exclusive clustering is used to reflect the fact that an object can simultaneously belong to more than one group (class). For instance, a person at a university can be both an enrolled student and an employee of the university. A non-exclusive clustering is also often used when, for example, an object is "between" two or more clusters and could reasonably be assigned to any of these clusters. Imagine a point halfway between two of the clusters of Figure 8.1. Rather than make a somewhat arbitrary assignment of the object to a single cluster, it is placed in all of the "equally good" clusters.

In a fuzzy clustering, every object belongs to every cluster with a membership weight that is between 0 (absolutely doesn't belong) and 1 (absolutely belongs). In other words, clusters are treated as fuzzy sets.. In fuzzy clustering, we often impose the additional constraint that the sum of the weights for each object must equal 1.) Similarly, probabilistic clustering has similar characteristics.

Complete versus Partial

A complete clustering assigns every object to a cluster, whereas a partial clustering does not.

For example, an application that uses clustering to organize documents for browsing needs to guarantee that all documents can be browsed

The motivation for a partial clustering is that some objects in a data set may not belong to well-defined groups.

Many times objects in the data set may represent noise, outliers, or "uninteresting background."

For example, some newspaper stories may share a common theme, such as global warming, while other stories are more generic or one-of-a-kind. Thus, to find the important topics in last month's stories, we may want to search only for clusters of documents that are tightly related by a common theme.

8.1.2Different Types of Clusters

Clustering aims to find useful groups of objects (clusters), where usefulness is defined by the goals of the data analysis

WellSeparated:.

A cluster is a set of objects in which each object is closer (or more similar) to every other object in the cluster than to any object not in the cluster.Sometimes a threshold is used to specify how close (or similar) to one another all the objects in a cluster.

This idealistic definition of a cluster is satisfied only when the data contains natural clusters that are quite far from each other.

Figure 8.2(a) gives an example of wellseparated clusters that consists of two groups of points in a two-dimensional space. The distance between any two points in different groups is larger than the distance between any two points within a group.

Well-separated clusters do not need to be globular, but can have any shape

Prototype-Based

A cluster is a set of objects in which each object is closer (more similar) to the prototype that defines the cluster than to the prototype of any other cluster.

For data with continuous attributes, the prototype of a cluster is often a centroid, i.e., the average (mean) of all the points in the cluster.

when the data has categorical attributes, the prototype is often a medoid, i.e., the most representative point of a cluster.

For many types of data, the prototype is the most central point, and in such instances, we commonly refer to prototype based clusters as center-based clusters , such clusters tend to be globular

Figure 8.2(b) shows an example of center-based clusters.

Graph-Based

Graph-Based If the data is represented as a graph, where the nodes are objects and the links represent connections among object **Graph-Based** If the data is represented as a graph, where the nodes are objects and the links represent connections among objects, then a cluster can be defined as a connected component; i.e., a group of objects that are connected to one another, but that have no connection to objects outside the group. An important example of graph-based clusters are contiguity-based clusters, where two objects are connected only if they are within a specified distance of each other. This implies that each object in a contiguity-based cluster is closer to some other object in the cluster than to any point in a different cluster.

Figure 8.2(c) shows an example of such clusters for two-dimensional points.

This definition of a cluster is useful when clusters are irregular or intertwined, but can have trouble when noise is present since, a small bridge of points can merge two distinct clusters.

Other types of graph-based clusters are also possible. One such approach defines a cluster as a clique i.e., a set of nodes in a graph that are completely connected to each other. Specifically, if we add connections between objects in the order of their distance from one another, a cluster is formed when a set of objects forms a clique.

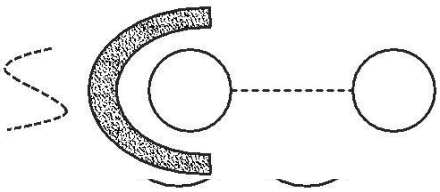
Density-Based A cluster is a dense region of objects that is surrounded by a region of low density. Figure 8.2(d) shows some density-based clusters for data created by adding noise to the data of Figure 8.2(c). The two circular clusters are not merged, as in Figure 8.2(c), because the bridge between them fades into the noise. Likewise, the curve that is present in Figure 8.2(c).

A density based definition of a cluster is often employed when the clusters are irregular or intertwined, and when noise and outliers are present.

Shared-Property (Conceptual Clusters) More generally, we can define a cluster as a set of objects that share some property. This definition encompasses all the previous definitions of a cluster; e.g., objects in a center-based cluster share the property that they are all closest to the same centroid or medoid. However, the shared-property approach also includes new types of clusters.

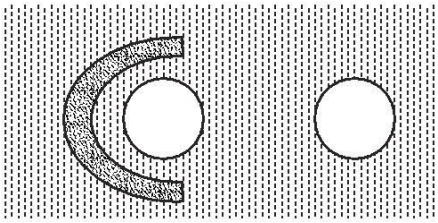
Consider the clusters shown in Figure 8.2(e). A triangular area (cluster) is adjacent to a rectangular one, and there are two intertwined circles (clusters). In both cases, a clustering algorithm would need a very specific concept of a cluster to successfully detect these clusters. The process of finding such clusters is called conceptual clustering.

(a) Well-separated clusters. Each point is closer to all of the points in its cluster than to any point in another cluster.

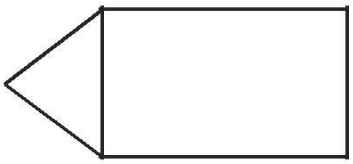


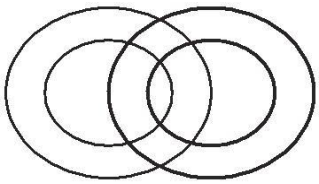
(b) Center-based clusters. Each point is closer to the center of its cluster than to the center of any other cluster.

c)Contiguity-based clusters. Each point is closer to at least one point in its cluster than to any point in another cluster



(d) Density-based clusters. Clus-ters are regions of high density sep-arated by regions of low density.





(e) Conceptual clusters. Points in a cluster share some general property that derives from the entire set of points. (Points in the intersection of the circles belong to both.)

Figure 8.2. Different types of clusters as illustrated by sets of two-dimensional points.

Road Map

In this chapter, we use the following three simple, but important techniques to introduce many of the concepts involved in cluster analysis.

- K-means. This is a prototype-based, partitional clustering technique that attempts to find a user-specified number of clusters (K), which are represented by their centroids.

- Agglomerative Hierarchical Clustering. This clustering approach refers to a collection of closely related clustering techniques that produce a hierarchical clustering by starting with each point as a singleton cluster and then repeatedly merging the two closest clusters until a single, all-encompassing cluster remains. Some of these techniques have a natural interpretation in terms of graph-based clustering, while others have an interpretation in terms of a prototype-based approach.

- DBSCAN. This is a density-based clustering algorithm that produces a partitional clustering, in which the number of clusters is automatically determined by the algorithm. Points in low-density regions are classified as noise and omitted; thus, DBSCAN does not produce a complete clustering.

8.2 K-means

Prototype-based clustering techniques create a one-level partitioning of the data objects. There are a number of such techniques, but two of the most prominent are K-means and K-medoid. K-means defines a prototype in terms of a centroid, which is usually the mean of a group of points.

8.2.1 The Basic K-means Algorithm

The K-means clustering technique is simple, and we begin with a description of the basic algorithm.

Steps:

1. We first choose K initial centroids, where K is a user-specified parameter, namely, the number of clusters desired.
2. Each point is then assigned to the closest centroid, and each collection of points assigned to a centroid is a cluster.
3. The centroid of each cluster is then updated based on the points assigned to the cluster.
4. We repeat the assignment and update steps until no point changes clusters, or equivalently, until the centroids remain the same.

Algorithm 8.1 Basic K-means algorithm.

Select K points as initial centroids.

1: repeat

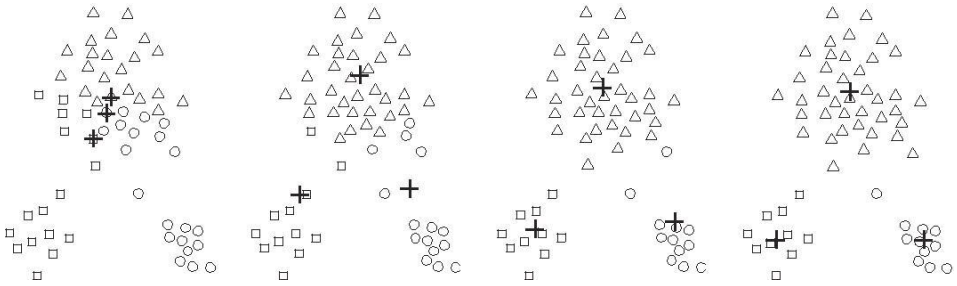
2: Form K clusters by assigning each point to its closest centroid.

3: Recompute the centroid of each cluster.

4: until Centroids do not change.

The operation of K-means is illustrated in Figure 8.3, which show, starting from three centroids

In the first step, shown in Figure 8.3(a), points are assigned to the initial centroids, which are all in the larger group of points. For this example, we use the mean as the centroid. After points are assigned to a centroid, the centroid is then updated again.



a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

Figure 8.3. Using the K-means algorithm to find three clusters in sample data.

. In steps 2, 3, and 4, which are shown in Figures 8.3 (b), (c), and (d), respectively, two of the centroids move to the two small groups of points at the bottom of the figures. When the K-means algorithm terminates in Figure 8.3(d), because no more changes occur, the centroids have identified the natural groupings of points.

We consider each of the steps in the basic K-means algorithm in more detail and then provide an analysis of the algorithm's space and time complexity.

Assigning Points to the Closest Centroid

To assign a point to the closest centroid, we need a proximity measure that quantifies the notion of "closest" for the specific data under consideration. Euclidean (L_2) distance is often used for data points in Euclidean space, while cosine similarity is more appropriate for documents. However, there may be several types of proximity measures that are appropriate for a given type of data. For example, Manhattan (L_1) distance can be used for Euclidean data, while the Jaccard measure is often employed for documents.

Table 8.1. Table of notation.

Symbol	Description
x	An object.
C_i	The i^{th} cluster.
c_i	The centroid of cluster C_i.
c	The centroid of all points.
m_i	The number of objects in the i^{th} cluster.
m	The number of objects in the data set.
K	The number of clusters.

Euclidean space, it is possible to avoid computing many of the similarities, thus significantly speeding up the K-means algorithm. Bisecting K-means (described in Section 8.2.3) is another approach that speeds up K-means by reducing the number of similarities computed.

Centroids and Objective Functions

- Step 4 of the K-means algorithm was stated rather generally as "recompute the centroid of each cluster," since the centroid can vary, depending on the proximity measure for the data and the goal of the clustering.
- The goal of the clustering is typically expressed by an objective function that depends on the proximities of the points to one another or to the cluster centroids; e.g., minimize the squared distance of each point to its closest centroid.
- However, the key point is this: once we have specified a proximity measure and an objective function, the centroid that we should choose can often be determined mathematically.

Data in Euclidean Space

Consider data whose proximity measure is Euclidean distance.

For our objective function, which measures the quality of a clustering, we use the sum of the squared error (SSE), which is also known as scatter.

In other words, we calculate the error of each data point, i.e., its Euclidean distance to the closest centroid, and then compute the total sum of the squared errors.

Given two different sets of clusters that are produced by two different runs of K-means, we prefer the one with the smallest squared error since this means that the prototypes (centroids) of this clustering

are a better representation of the points in their cluster

Using the notation in Table 8.1, the SSE is formally defined as follows:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i, x)^2 \quad (8.1)$$

where dist is the standard Euclidean (L_2) distance between two objects in Euclidean space.
Given these assumptions, it can be shown (see Section 8.2.6) that the centroid that minimizes the SSE of the cluster is the mean. Using the notation in Table 8.1, the centroid (mean) of the i^{th} cluster is defined by Equation 8.2.

$$Ci = \frac{1}{mi} \sum_{x \in Ci} x$$

To illustrate, the centroid of a cluster containing the three two-dimensional points, (1,1), (2,3), and (6,2), is ((1 + 2 + 6)/3, ((1 + 3 + 2)/3) = (3, 2).

If there are two different sets of clusters are produced by two different K-means algorithm and that are guaranteed to converge. Table 8.2 shows some possible choices.

Table 8.2. K-means: Common choices for proximity, centroids, and objective functions.

Proximity Function	Centroid	Objective Function
Manhattan (L_1)	median	Minimize sum of the L_1 distance of an ob- ject to its cluster centroid
Squared Euclidean (L_2^2)	mean	Minimize sum of the squared L_2 distance of an object to its cluster centroid
cosine	mean	Maximize sum of the cosine similarity of an object to its cluster centroid
Bregman divergence	mean	Minimize sum of the Bregman divergence of an object to its cluster centroid

Document Data

We even consider document data other than Euclidean data and the cosine similarity measure. Here we assume document-term matrix, our objective is to maximize the similarity of the documents in a cluster to the cluster centroid; this quantity is known as the **cohesion of the cluster.** The analogous

quantity to the total SSE is the total cohesion, which is given by Equation 8.3.

$$\textbf{Total Cohesion}=\sum_{i=1}^{k}\sum_{x\in Ci} cosine(x, Ci) \quad \text{.....8.3}$$

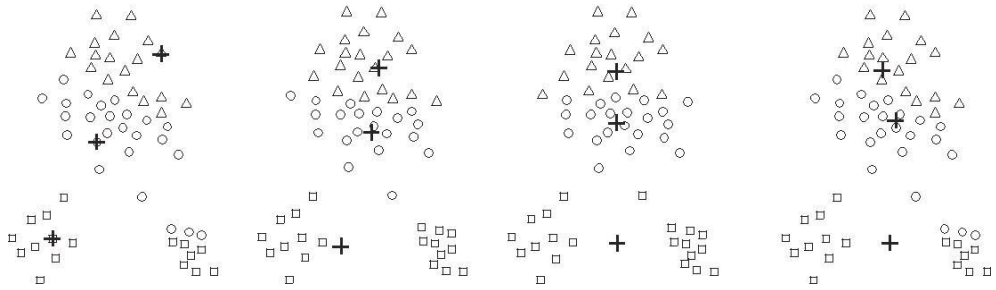
Choosing Initial Centroids

When random initialization of centroids is used, different runs of K-means typically produce different total SSEs. We illustrate this with the set of two-dimensional points shown in Figure 8.3, which has three natural clusters of points. Figure 8.4(a) shows a clustering solution that is the global minimum of the SSE for three clusters, while Figure 8.4(b) shows a suboptimal clustering that is only a local minimum.

(a) Optimal clustering. (b) Suboptimal clustering.

Figure 8.4. Three optimal and non-optimal clusters.

Choosing the proper initial centroids is the key step of the basic K-means procedure. A common approach is to choose the initial centroids randomly, but the resulting clusters are often poor.

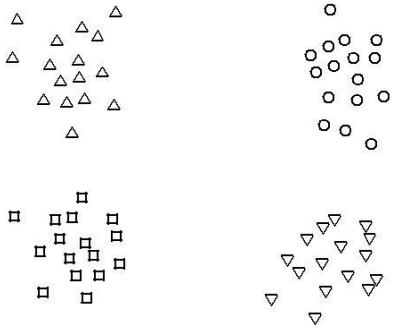


a.)Iteration 1 (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

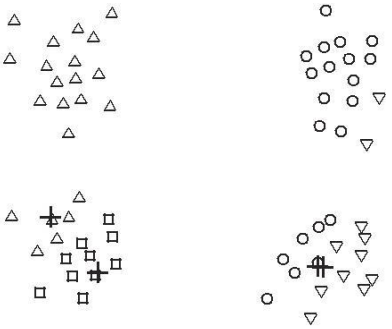
Figure 8.5. Poor starting centroids for K-means.

cluster, the centroids will redistribute themselves so that the "true" clusters are found. However, Figure 8.7 shows that if a pair of clusters has only one initial centroid and the other pair has three, then two of the true clusters will be combined and one true cluster will be split.

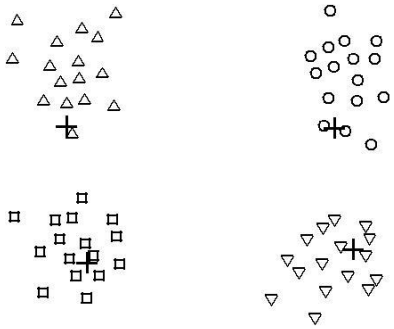
Note that an optimal clustering will be obtained as long as two initial centroids fall anywhere in a pair of clusters, since the centroids will redistribute themselves, one to each cluster. Unfortunately, as the number of clusters becomes larger, it is increasingly likely that at least one pair of clusters will have only one initial centroid.



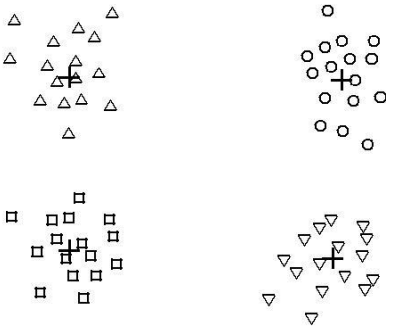
(a) Initial points.



(b) Iteration 1.



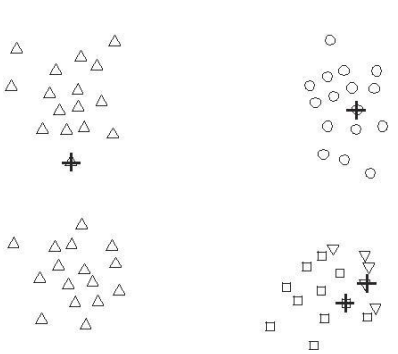
(c) Iteration 2.



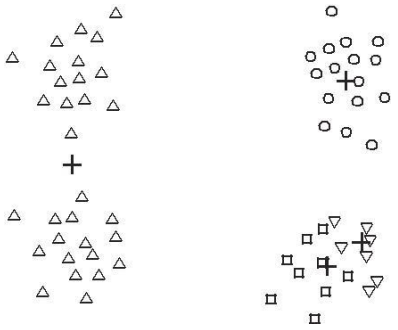
(d) Iteration 3.

Figure 8.6. Two pairs of clusters with a pair of initial centroids within each pair of clusters.

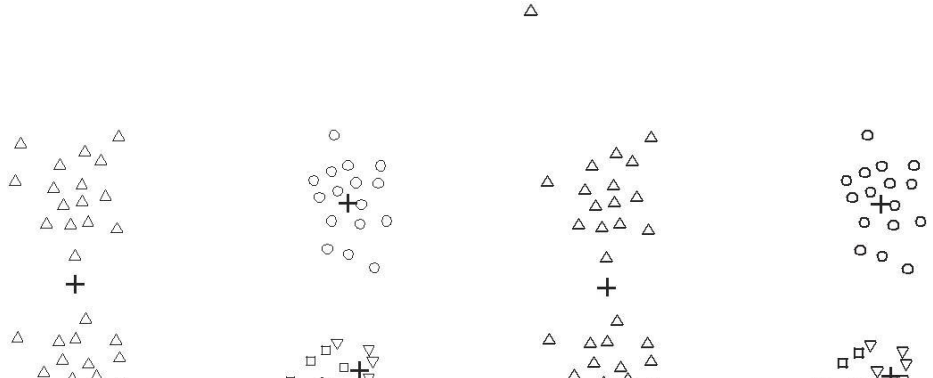
Set of centroids that is guaranteed to be not only randomly selected but also well separated. Unfortunately, such an approach can select outliers, rather than points in dense regions (clusters). Also, it is expensive to compute the farthest point from the current set of initial centroids. To overcome these problems, this approach is often applied to a sample of the points. Since outliers are rare, they tend not to show up in a random sample.



(a) Iteration 1.



(b) Iteration 2.



(c) Iteration 3. (d) Iteration 4.

Time and Space Complexity

The space requirements for K-means are modest because only the data points and centroids are stored. Specifically, the storage required is O((m + K)n), where m is the number of points and n is the number of attributes. The time requirements for K-means are also modest—basically linear in the number of data points. In particular, the time required is $O(I * K * m * n)$, where I is the number of iterations required for convergence.

8.2.2 K-means: Additional Issues

Handling Empty Clusters

One of the problems with the basic K-means algorithm given earlier is that empty clusters can be obtained if no points are allocated to a cluster during the assignment step. If this happens, then a strategy is needed to choose a replacement centroid, since otherwise, the squared error will be larger than necessary. One approach is to choose the point that is farthest away from any current centroid. If nothing else, this eliminates the point that currently contributes most to the total squared error. Another approach is to choose the replacement centroid from the cluster that has the highest SSE. This will typically split the cluster and reduce the overall SSE of the clustering. If there are several empty clusters, then this process can be repeated several times.

Outliers

When the squared error criterion is used, outliers can unduly influence the clusters that are found. In particular, when outliers are present, the resulting cluster centroids (prototypes) may not be as representative as they otherwise would be and thus, the SSE will be higher as well. Because of this, it is often useful to discover outliers and eliminate them beforehand. It is important, however, to appreciate that there are certain clustering applications for which outliers should not be eliminated. When clustering is used for data com-pression, every point must be clustered, and in some cases, such as financial analysis, apparent outliers, e.g., unusually profitable customers, can be the most interesting points.

Reducing SSE with postprocessing

Two strategies that decrease the total SSE by increasing the number of clusters are the following:

Split a cluster: The cluster with the largest SSE is usually chosen, but we could also split the cluster with the largest standard deviation for one particular attribute.

Introduce a new cluster centroid: Often the point that is farthest from any cluster center is chosen. We can easily determine this if we keep track of the SSE contributed by each point. Another approach is to choose randomly from all points or from the points with the highest SSE.

Two strategies that decrease the number of clusters, while trying to mini-mize the increase in total SSE, are the following:

Disperse a cluster: This is accomplished by removing the centroid that cor-responds to the cluster and reassigning the points to other clusters. Ide-ally, the cluster that is dispersed should be the one that increases the total SSE the least.

Merge two clusters: The clusters with the closest centroids are typically chosen, although another, perhaps better, approach is to merge the two clusters that result in the smallest increase in total SSE. These two merging strategies are the same ones that are used in the hierarchical

Bisecting K-means

The bisecting K-means algorithm is a straightforward extension of the basic K-means algorithm that is based on a simple idea: to obtain K clusters, split the set of all points into two clusters, select one of these clusters to split, and

Algorithm 8.2 Bisecting K-means algorithm.

1: Initialize the list of clusters to contain the cluster consisting of all points.

2: repeat

3: Remove a cluster from the list of clusters.

4: {Perform several “trial” bisections of the chosen cluster.}

5: for i = 1 to number of trials do

6: Bisect the selected cluster using basic K-means.

7: end for

8: Select the two clusters from the bisection with the lowest total SSE.

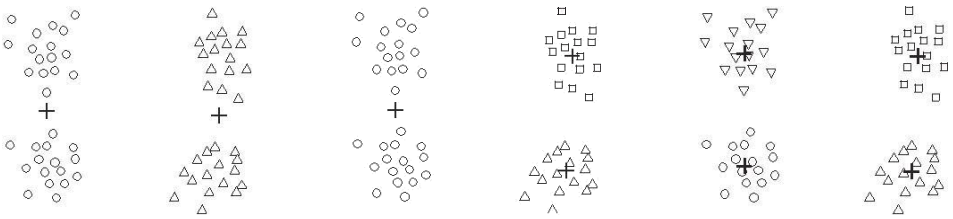
9: Add these two clusters to the list of clusters.

10: until Until the list of clusters contains K clusters.

There are a number of different ways to choose which cluster to split. We can choose the largest cluster at each step, choose the one with the largest SSE, or use a criterion based on both size and SSE. Different choices result in different clusters.

We often refine the resulting clusters by using their centroids as the initial centroids for the basic K-means algorithm. This is necessary because, although the K-means algorithm is guaranteed to find a clustering that represents a local minimum with respect to the SSE, in bisecting K-means we are using the K-means algorithm "locally," i.e., to bisect individual clusters. Therefore, the final set of clusters does not represent a clustering that is a local minimum with respect to the total SSE.

Finally, by recording the sequence of clusterings produced as K-means bisects clusters, we can also use bisecting K-means to produce a hierarchical clustering.



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3.

Figure 8.8. Bisecting K-means on the four clusters example.

8.2.4 K-means and Different Types of Clusters

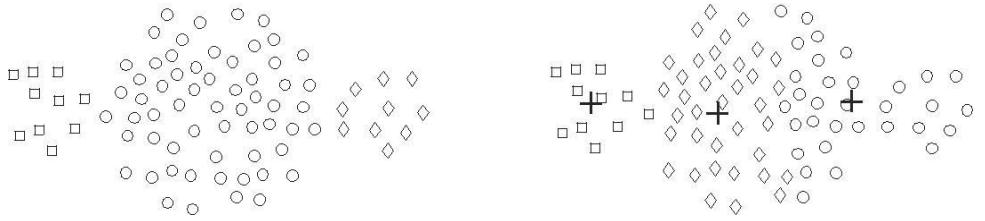
K-means and its variations have a number of limitations with respect to finding different types of clusters. In particular, K-means has difficulty detecting the "natural" clusters, when clusters have non-spherical shapes or widely different sizes or densities. This is illustrated by Figures 8.9, 8.10, and 8.11. In Figure 8.9, K-means cannot

find the three natural clusters because one of the clusters is much larger than the other two, and hence, the larger cluster is broken, while one of the smaller clusters is combined with a portion of the larger cluster. In Figure 8.10, K-means fails to find the three natural clusters because the two smaller clusters are much denser than the larger cluster. Finally, in Figure 8.11, K-means finds two clusters that mix portions of the two natural clusters because the shape of the natural clusters is not globular.

The difficulty in these three situations is that the K-means objective func-tion is a mismatch for the kinds of clusters we are trying to find since it is minimized by globular clusters of equal size and density or by clusters that are well separated. However, these limitations can be overcome, in some sense, if the user is willing to accept a clustering that breaks the natural clusters into a number of subclusters. Figure 8.12 shows what happens to the three previous data sets if we find six clusters instead of two or three. Each smaller cluster is pure in the sense that it contains only points from one of the natural clusters.

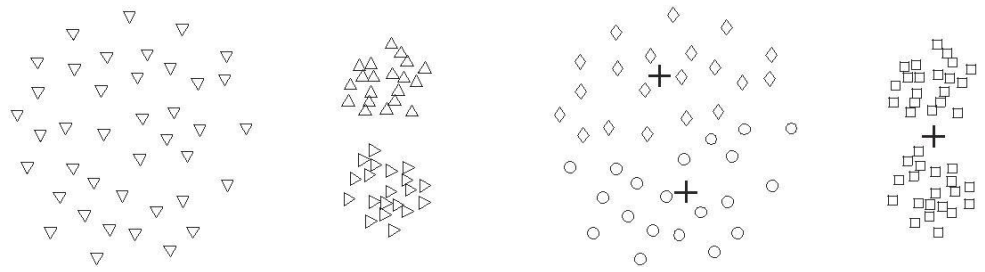
8.2.5 Strengths and Weaknesses

K-means is simple and can be used for a wide variety of data types. It is also quite efficient, even though multiple runs are often performed. Some variants, including bisecting K-means, are even more efficient, and are less susceptible to initialization problems. K-means is not suitable for all types of data,



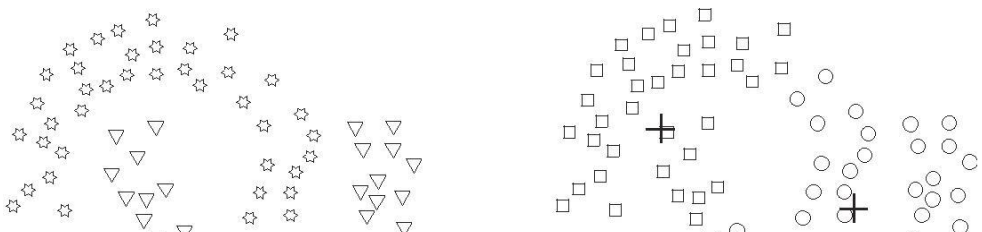
(a) Original points. (b) Three K-means clusters.

Figure 8.9. K-means with clusters of different size.



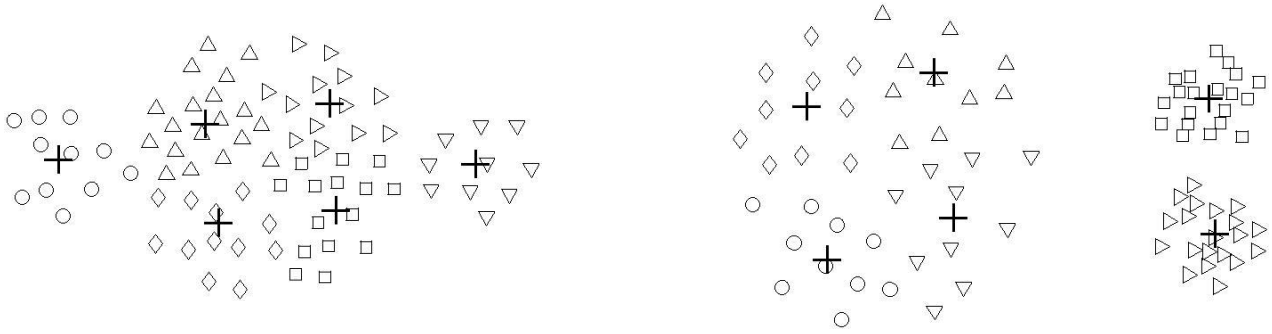
(a) Original points. (b) Three K-means clusters.

Figure 8.10. K-means with clusters of different density.

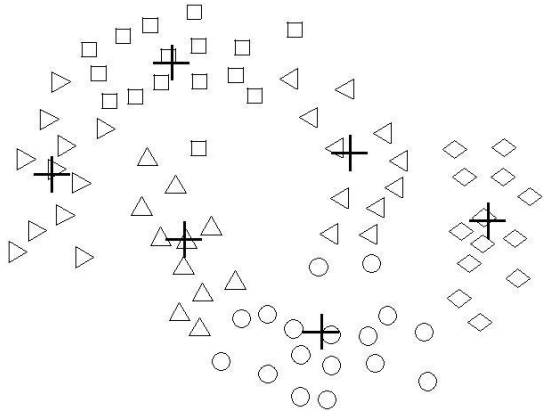


(a) Original points. (b) Two K-means clusters.

Figure 8.11. K-means with non-globular clusters.



(a) Unequal sizes. (b) Unequal densities.



(c) Non-spherical shapes.

Figure 8.12. Using K-means to find clusters that are subclusters of the natural clusters.

8.2 K-means

however. It cannot handle non-globular clusters or clusters of different sizes and densities, although it can typically find pure subclusters if a large enough number of clusters is specified. K-means also has trouble clustering data that contains outliers. Outlier detection and removal can help significantly in such situations. Finally, K-means is restricted to data for which there is a notion of a center (centroid). A related technique, K-medoid clustering, does not have this restriction, but is more expensive.

8.2.6 K-means as an Optimization Problem

As mentioned earlier, given an objective function such as "minimize SSE,' clustering can be treated as an optimization problem.

- One technique, which is known as gradient descent, is based on picking an initial solution
- and then repeating the following two steps: compute the change to the solution that best optimizes the objective function and then update the solution. We assume that the data is one-dimensional,i. e., dist(x,y) : (x - y)^2.

Derivation of K-means as an Algorithm to Minimize the SSE

- We show how the Problem here is when the proximity function is Euclidean distance and the objective is to minimize the SSE is given how the centroid for the K-means algorithm can be mathematically derived .
- Specifically,we investigate how we can best update a cluster centroid so that the cluster SSE is minimized.

$$SAE = \sum_{i=1}^{k} \sum_{x \in Ci} dist_{L1}(Ci, x)$$

We can solve for the kth centroid ,w hich minimizes above Equation , by differentiating the SAE, setting it equal to 0, and solving

$$\begin{aligned} \frac{\partial}{\partial c_k}\text{SAE} &= \frac{\partial}{\partial c_k}\sum_{i=1}^{K}\sum_{x \in C_i}|c_i - x| \\ &= \sum_{i=1}^{K}\sum_{x \in C_i}\frac{\partial}{\partial c_k}|c_i - x| \\ &= \sum_{x \in C_k}\frac{\partial}{\partial c_k}|c_k - x| = 0 \end{aligned}$$

$$\sum_{x \in C_k}\frac{\partial}{\partial c_k}|c_k - x| = 0 \Rightarrow \sum_{x \in C_k} sign(x - c_k) = 0$$

If we solve for c_k, we find that c_k: med'ian{x € C_k }, the median of the points in the cluster. The median of a group of points is straightforward to compute and less susceptible to distortion by outliers.

8.3 Agglomerative Hierarchical Clustering

Hierarchical clustering techniques are a second important category of clustering methods. There are two basic approaches for generating a hierarchical clustering:

Agglomerative: Start with the points as individual clusters and, at each step, merge the closest pair of clusters. This requires defining a notion of cluster proximity.

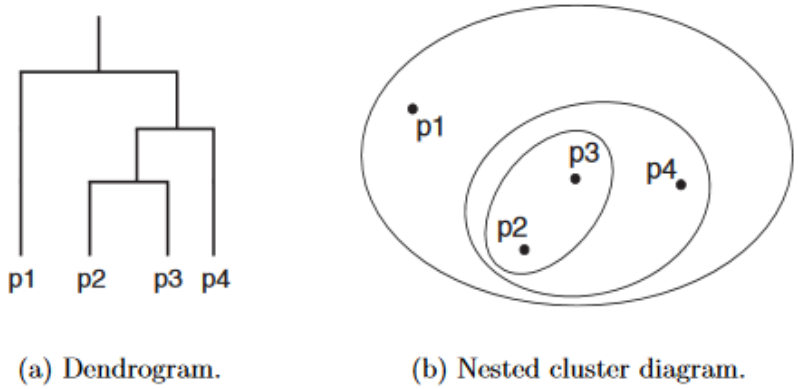
Divisive: Start with one, all-inclusive cluster and, at each step, split a cluster until only singleton

clusters of individual points remain. In this case) we need to decide which cluster to split at each step and how to do the splitting.

A hierarchical clustering is often displayed graphically using a tree-like diagram called a dendrogram, which displays both the cluster-subcluster relationships and the order in which the clusters were merged (agglomerative view) or split (divisive view). For sets of two-dimensional points, a hierarchical clustering can also be graphically represented using a nested cluster diagram.

8.3.1 Basic Agglomerative Hierarchical Clustering Algorithm

Many agglomerative hierarchical clustering techniques are variations on a single approach: starting with individual points as clusters, successively merge the two closest clusters until only one cluster remains.



(a) Dendrogram. (b) Nested cluster diagram.

Figure 8.13. A hierarchical clustering of four points shown as a dendrogram and as nested clusters.

8.3.1 Basic Agglomerative Hierarchical Clustering Algorithm

Many agglomerative hierarchical clustering techniques are variations on a single approach: starting with individual points as clusters, successively merge the two closest clusters until only one cluster remains. This approach is expressed more formally in Algorithm 8.3.

Algorithm 8.3

Basic agglomerative hierarchical clustering algorithm.

1: Compute the proximity matrix, if necessary.
2: repeat
3: Merge the closest two clusters.
4: Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
5: until only one cluster remain

Step1:Defining Proximity between Clusters

Cluster proximity is typically defined with a particular type of cluster in mind For example, many **agglomerative hierarchical clustering techniques, such as MIN, MAX, and Group Average,** come from a graph-based view of clusters.

MIN(single Link) defines cluster proximity as the proximity between the closest two points that are in different clusters, or using graph terms, the shortest edge between two nodes in different subsets of nodes.

MAX(Complete link) takes the proximity between the farthest two points in different clusters to be the cluster proximity, or using graph terms, the longest edge between two nodes in different subsetsof nodes.

Group average technique, definescluster proximity to be the average pairwise proximities of all pairs of points from different clusters.

Figure 8.14 illustrates these three approaches.

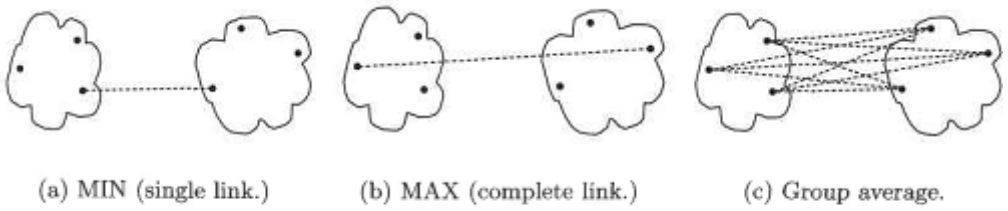


Figure 8.14. Graph-based definitions of cluster proximity

An alternative technique, Ward's method, also assumes that a cluster is represented by its centroid, but it measures the proximity between two clusters in terms of the increase in the SSE that results from merging the two clusters.

Time and Space Complexity

The basic agglomerative hierarchical clustering algorithm just presented uses a proximity matrix. This requires the storage **of** $\frac{1}{2}m^2$ proximities (assuming the proximity matrix is symmetric) where m is the number of data points. Hence,the total space complexity is $O(m^2)$. Hence space complexity is **$O(m^2)$.**

The overall time required for a hierarchical clustering based on Algorithm 8.3 is **$O(m^2 \log m)$.**

8.3.2 Specific Techniques(SINGLE,COMPLETE,GROUP AVERAGE)

To illustrate the behavior of the various hierarchical clustering algorithms, we shall use sample data that consists of 6 two-dimensional points, which are shown in Figure 8.15. The x and y coordinates of the points and the Euclidean distances between them are shown in Tables 8.3 and 8.4. respectivelv.

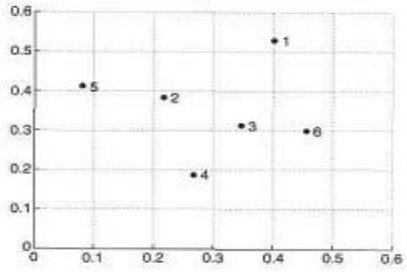


Figure 8.15. Set of 6 two-dimensional points.

Point	x Coordinate	y Coordinate
p1	0.40	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.30

Table 8.3. xy coordinates of 6 points.

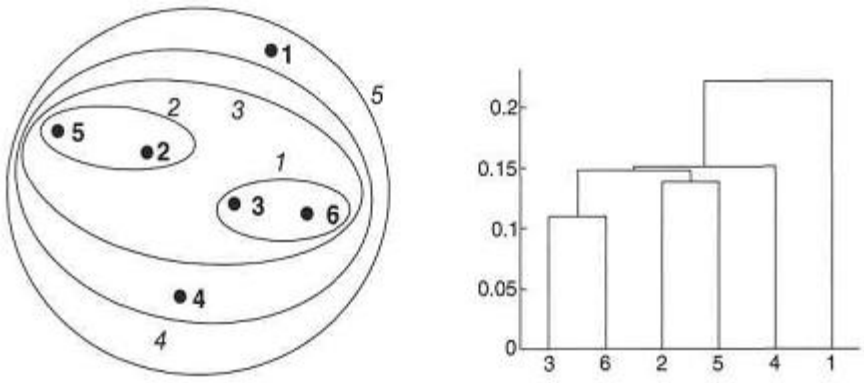
Table 8.4 EUCLIDEAN DISTANCE MATRIX FOR 2 POINTS

	P1	P2	P3	P4	P5	P6
P1	**0**					
P2	**0.23**	**0**				
P3	**0.22**	**0.15**	**0**			
P4	**0.37**	**0.20**	**0.15**	**0**		
P5	**0.34**	**0.14**	**0.28**	**0.29**	**0**	
P6	**0.23**	**0.25**	**0.11**	**0.22**	**0.39**	**0**

(Refer The Problem Solved In Class)

- **single Link or MIN**

- For the single link or MIN version of hierarchical clustering, the proximity of two clusters is defined as the minimum of the distance (maximum of the similarity) between any two points in the two different clusters.
- Initially we represent all points as singleton clusters. Our goal is to group all clusters, such that end of the iterations, we have all points in single cluster. Clustering is based on the minimum distance between the objects of the two clusters.
- The single link technique is good at handling non-elliptical shapes,but is sensitive to noise and outliers.



(a) Single link clustering. (b) Single link dendrogram.

Figure 8.16. Single link clustering of the six points shown in Figure 8.15.

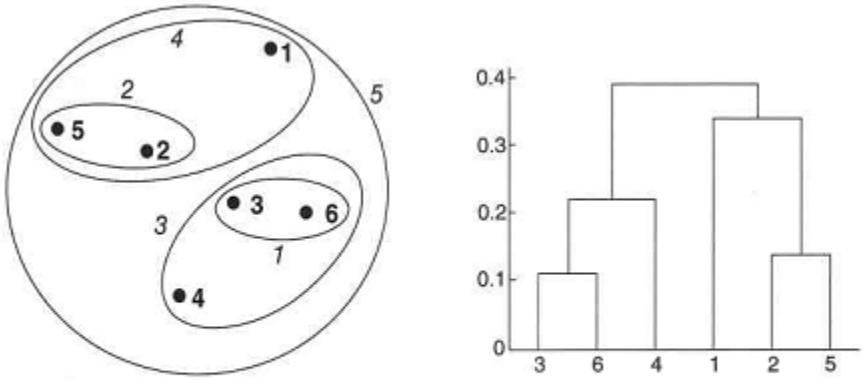
We start clustering 3&6 in the first step, whose distance is minimum in the matrix i.e, 0.11. After the cluster formation , need to update the proximity matrix with clustered points .similarly other clusters are formed based on the updated matrix. This repeats until only one cluster remains.

$$\begin{aligned} dist(\{3,6\},\{2,5\}) &= \min(dist(3,2), dist(6,2), dist(3,5), dist(6,5)) \\ &= \min(0.15, 0.25, 0.28, 0.39) \\ &= 0.15. \end{aligned}$$

- **Complete Link or MAX or CLIQUE**

- For the complete link or MAX version of hierarchical clustering, the proximity of two clusters is defined as the maximum of the distance (minimum of the similarity) between any two points in the two different clusters.
- Initially we represent all points as singleton clusters. Our goal is to group all clusters, such that end of the iterations, we have all points in single cluster. Clustering is based on the shortest distance between the objects of the two clusters. Then a group of points is not a cluster until all the points in it are completely linked, i.e., form a clique.
- Complete link is less susceptible to noise and outliers, but it can break large clusters and it favors globular shapes.

Figure 8.17 showsthe results of applying MAX to the sample data set of six points



(a) Complete link clustering. (b) Complete link dendrogram.

Figure 8.17. Complete link clustering of the six points shown in Figure 8.15.

Points 3 and 6 are merged first. However, {3,6} is merged with {4}, instead of {2,5} or {1}

$$\begin{aligned} dist(\{3,6\},\{4\}) &= \max(dist(3,4), dist(6,4)) \\ &= \max(0.15, 0.22) \\ &= 0.22. \\ dist(\{3,6\},\{2,5\}) &= \max(dist(3,2), dist(6,2), dist(3,5), dist(6,5)) \\ &= \max(0.15, 0.25, 0.28, 0.39) \\ &= 0.39. \\ dist(\{3,6\},\{1\}) &= \max(dist(3,1), dist(6,1)) \\ &= \max(0.22, 0.23) \\ &= 0.23. \end{aligned}$$

Group Average

The proximity of two clusters is defined as the average pairwise proximity among all pairs of points in the different clusters. This is an intermediate approach between the single and complete link approaches. Thus, for group average, the cluster proximity proximity(Ci,Cj) of clusters Ci and Cj, which are of size mi and mj, respectively, is expressed by the following equation:

$$proximity(C_i, C_j) = \frac{\sum_{\substack{\mathbf{x} \in C_i \\ \mathbf{y} \in C_j}} proximity(\mathbf{x}, \mathbf{y})}{m_i * m_j}. \qquad (8.6)$$

To illustrate how group works, we calculate the distance between some clusters average

$$
\begin{aligned}
dist(\{3,6,4\},\{1\}) &= (0.22+0.37+0.23)/(3*1) \\
&= 0.28 \\
dist(\{2,5\},\{1\}) &= (0.2357+0.3421)/(2*1) \\
&= 0.2889 \\
dist(\{3,6,4\},\{2,5\}) &= (0.15+0.28+0.25+0.39+0.20+0.29)/(6*2) \\
&= 0.26
\end{aligned}
$$

Because $dist(\{3,6,4\},\{2,5\})$ is smaller than $dist(\{3,6,4\},\{1\})$ and $dist(\{2,5\},\{1\})$, clusters $\{3,6,4\}$ and $\{2,5\}$ are merged at the fourth stage. ■

Ward's Method and Centroid Methods

For Ward's method, the proximity between two clusters is defined as the increase in the squared error that results when two clusters are merged. Thus, this method usesthe same objective function as K-means clustering. Ward's method is very similar to the group average method when the proximity between two points is taken to be the square of the distance between them.

Centroid methods calculate the proximity between two clusters by calculating the distance between the centroids of clusters.

8.3.3 Key Issues in Hierarchical Clustering

1. Lack of a Global Objective Function

- Agglomerative hierarchical clustering techniques use various criteria to decide locally, at each step, which clusters should be merged (or split for divisive approaches).
- This approach avoid the difficulty of attempting to solve a hard combinatorial optimization problem.

2. Ability to Handle Different Cluster Sizes

- How to treat the relative sizes of the pairs of clusters that are merged. There are two approaches: weighted, which treats all clusters equally, and unweighted, which takes the number of points in each cluster into account.

3. Merging Decisions Are Final

- Agglomerative hierarchical clustering algorithms tend to make good local decisions about combining two clusters which makes use of pairwise similarity of all points.
- However, once a decision is made to merge two clusters, it cannot be undone at a later time. This approach prevents a local optimization criterion from becoming a global optimization criterion.

Agglomerative hierarchical clustering algorithms are expensive in terms of their computational and storage requirements. The fact that all merges are final can also cause trouble for noisy, high-dimensional data, such as document data.

8.4 DBSCAN

Density-based clustering locates regions of high density that are separated from one another by regions of low density. DBSCAN is a simple and effective density-based clustering algorithm .

8.4.1 Traditional Density: Center-Based Approach

In the center-based approach, density is estimated for a particular point in the data set by counting the nurnber of points within a specified radius,Eps,of that point. This includes the point itself. This technique is graphically illustrated by Figure 8.20. The number of points within a radius of Eps of point A is 7, including A itself.

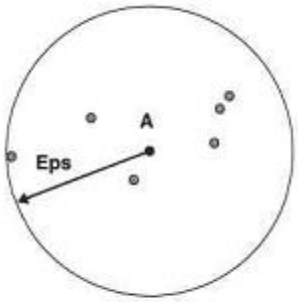


Figure 8.20. Center-based density.

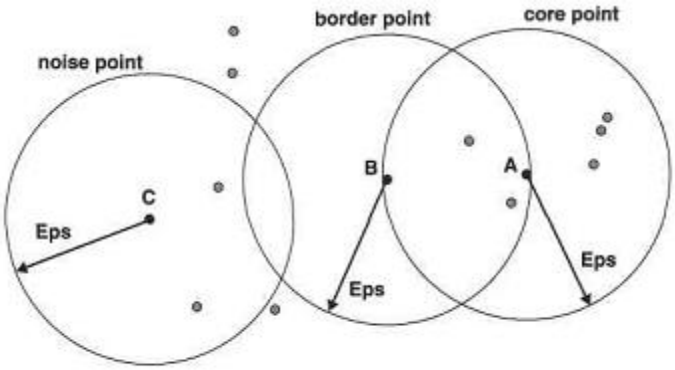


Figure 8.21. Core, border, and noise points.

Classification of Points According to Center-Based Density

The center-based approach to density allows us to classify a point as being

Core points: These points are in the interior of a density-based cluster. A point is a core point if the number of points within a given neighbourhood i.e, around the point's radius(Eps), exceeds a certain threshold, MinPts, which is a user-specified parameter. In Figure 8.21, point A is a core point, for the indicated radius (Eps) 1f MinPts >= 7.

Border points: A border point is not a core point, but falls within the neighborhood of a core point. In Figure 8.21, point B is a border point. A border point can fall within the neighbourhoods of several core points.

Noise points(Background point): A noise point is any point that is neither a core point nor a

border point. In Figure 8.21, point C is a noise point.

8.4.2 The DBSCAN Algorithm

Given the previous definitions of core points, border points, and noise points, the DBSCAN algorithm can be informally describedas follows. Any two core points that are close enough-within a distance Eps of one another are put in the same cluster. Likewise, any border point that is close enough to a core point is put in the same cluster as the core point. Noise points are discarded.

The formal details are given in Algorithm. 8.4.

Algorithm 8.4 DBSCAN algorithm.

1: Label all points as core, border, or noise points.
2: Eliminate noise points,
3: Put an edge between all core points that are within Eps of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.

Time and Space Complexity

- The basic time complexity of the DBSCAN algorithm is O(m x time to find points in the -Eps-neighborhood),where m is the number of points.
- In the worst case, this complexity is O(m^{2}).
- In low dimensional spaces the time complexity can be as low as O(m logm)
- The space requirement of DBSCAN, even for high-dimensional data, is O(m)

Strengths and Weaknesses

- It is relatively resistant to noise and can handle clusters of arbitrary shapes and sizes.
- Thus, DBSCAN can find many clusters that could not be found using K-means, such as those in Figure 8.22.
- DBSCAN has trouble when the clusters have widely varying densities.
- It also has trouble with high-dimensional data because density is more difficult to define for such data. One possible approach to dealing with such issues is SNN density based clustering.
- Finally, DBSCAN can be expensive when the computation of nearest neighbors requires computing all pairwise proximities, as is usually the case for high-dimensional data.

8.5 Cluster Evaluation

why cluster evaluation is necessary?

Each clustering algorithm defines its own type of cluster it may seem that each situation might require a different evaluation measure. For instance, K-means clusters might be evaluated in terms of the SSE, but for density-based clusters, which need not be globular, SSE would not work well at all.

8.5.1 Overview

Being able to distinguish whether there is non-random structure in the data is just one important aspect of cluster validation. The following is a list of several important issues for cluster validation.

1. Determining the clustering tendency of a set of data, i.e., distinguishing whether non-random structure actually exists in the data.
2. Determining the correct number of clusters.
3. Evaluating how well the results of a cluster analysis fit the data wi,thout referenceto external information.
4. Comparing the results of a cluster analysis to externally known results, such as externally provided class labels.
5. Comparing two sets of clusters to determine which is better.

The evaluation measures, or indices, that are applied to judge various aspects of cluster validity are traditionally **classified into the following three types.**

Unsupervised.
Measures the goodness of a clustering structure without respect to external information. An example of this is the SSE. Unsupervised measures of cluster validity are often further divided into two (compactness, tightness which determine how closely related the objects in a cluster are, and measures of cluster separation (isolation), which determine how distinct or well-separated a cluster is from other clusters. Unsupervised measures are often called internal indices because thev use only information present in the data set.

Supervised. Measures the extent to which the clustering structure discovered by a clustering algorithm rnatches some external structure. An example of a supervised index is entropy, which measures how well cluster labels match externally supplied class labels. Supervised measures are often called external indices Lrecause they use information not present in the data set.

Relative. Compares different clusterings or clusters. A relative cluster evaluation measure is a supervised or unsupervised evaluation measure that is used for the purpose of comparison. Thus, relative measures are not actually a separate type of cluster evaluation measure, but are instead a specific use of such measures. As an example, two K-means clusterings can be compared using either the SSE or entropy.

8.5.2 Unsupervised Cluster Evaluation Using Cohesion and Separation

In general, we can consider expressing overall cluster validity for a set of K clusters as a weighted sum of the validity of individual clusters,

$$\boldsymbol{Overall\ Validity} = \sum_{i=1}^{k} \boldsymbol{w_i}\ \boldsymbol{Validity}(\boldsymbol{C_i}) \quad \ldots\ldots\ldots \qquad \textbf{(8.8)}$$

The ***Validity*** function can be cohesion, separation, or some combination of these quantities. The weights will vary depending on the cluster validity measure.

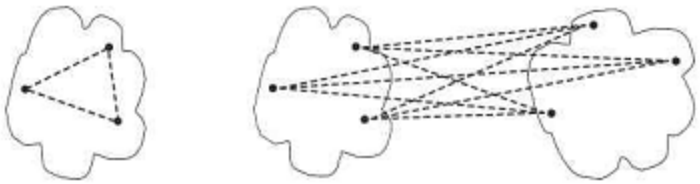
Graph-Based View of Cohesion and Separation

- For graph-based clusters, the **cohesion** of a cluster can be defined as the sum of the weights of the links in the proximity graph that connect points within the cluster.
- Likewise, the separation between two clusters can be measured by the sum of the weights of the links from points in one cluster to points in the other cluster. This is illustrated in Figure 8.27(b).

Mathematically, cohesion and separation for a graph-based cluster can be expressed using Equations 8.9 and 8.10, respectively. The proximity function can be a similarity, a dissimilarity, or a simple function of these quantities.

$$cohesion(C_i) = \sum_{\substack{\mathbf{x} \in C_i \\ \mathbf{y} \in C_i}} proximity(\mathbf{x}, \mathbf{y}) \quad (8.9)$$

$$separation(C_i, C_j) = \sum_{\substack{\mathbf{x} \in C_i \\ \mathbf{y} \in C_j}} proximity(\mathbf{x}, \mathbf{y}) \quad (8.10)$$



(a) Cohesion. (b) Separation.

Figure 8.27. Graph-based view of cluster cohesion and separation.

Prototype-Based View of Cohesion and Separation

For prototype-based clusters, the cohesion of a cluster can be defined as the sum of the proximities with respect to the prototype (centroid or medoid) of the cluster. Similarly, the separation between two clusters can be measured by the proximity of the two cluster prototypes.

$$cohesion(C_i) = \sum_{\mathbf{x} \in C_i} proximity(\mathbf{x}, \mathbf{c}_i) \quad (8.11)$$

$$separation(C_i, C_j) = proximity(\mathbf{c}_i, \mathbf{c}_j) \quad (8.12)$$

$$separation(C_i) = proximity(\mathbf{c}_i, \mathbf{c}) \quad (8.13)$$

Overall Measures of Cohesion and Separation

Table 8.6. Table of graph-based cluster evaluation measures.

Name	Cluster Measure	Cluster Weight	Type
$\mathcal{I}_1$	$\sum_{\substack{\mathbf{x}\in C_i \\ \mathbf{y}\in C_i}} proximity(\mathbf{x},\mathbf{y})$	$\frac{1}{m_i}$	graph-based cohesion
$\mathcal{I}_2$	$\sum_{\mathbf{x}\in C_i} proximity(\mathbf{x},\mathbf{c}_i)$	1	prototype-based cohesion
$\mathcal{E}_1$	$proximity(\mathbf{c}_i,\mathbf{c})$	m_i	prototype-based separation
$\mathcal{G}_1$	$\sum_{\substack{j=1 \\ j\neq i}}^{k} \sum_{\substack{\mathbf{x}\in C_i \\ \mathbf{y}\in C_j}} proximity(\mathbf{x},\mathbf{y})$	$\frac{1}{\sum_{\substack{\mathbf{x}\in C_i \\ \mathbf{y}\in C_i}} proximity(\mathbf{x},\mathbf{y})}$	graph-based separation and cohesion

Relationship between Prototype-Based Cohesion and Graph-Based Cohesion

$$\text{Cluster SSE} = \sum_{\mathbf{x}\in C_i} dist(\mathbf{c}_i,\mathbf{x})^2 = \frac{1}{2m_i}\sum_{\mathbf{x}\in C_i}\sum_{\mathbf{y}\in C_i} dist(\mathbf{x},\mathbf{y})^2 \qquad (8.14)$$

While the graph-based and prototype-based approaches to measuring the cohesion and separation of a cluster seem distinct, for some proximity measures they are equivalent.

Two Approaches to Prototype-Based Separation

- When proximity is measured by Euclidean distance, the traditional measure of separation between clusters is the between group sum of squares (SSB), which is the sum of the squared distance of a cluster centroid, $\mathbf{c}_i$, to the overall mean $\mathbf{c}$, of all the data points.
- By summing the SSB over all clusters, we obtain the total SSB, which is given by Equation 8.15, where $\mathbf{c}_i$, is the mean of the ith cluster and c is the overall mean.
- The higher the total SSB of a clustering, the more separated the clusters are from one another.

$$\text{Total SSB} = \sum_{i=1}^{K} m_i\, dist(\mathbf{c}_i,\mathbf{c})^2 \qquad \textbf{…….8.15}$$

The total SSB is directly related to the pairwise distances between the centroids.

$$\text{Total SSB} = \frac{1}{2K}\sum_{i=1}^{K}\sum_{j=1}^{K}\frac{m}{K}\, dist(\mathbf{c}_i,\mathbf{c}_j)^2 \qquad \textbf{----------8.16}$$

If in case, the cluster sizes are equal, then this relationship takes the simple form given by Equation 8.16.

Relationship between Cohesion and Separation

In some cases, there is also a strong relationship between cohesion and separation. Specifically, it is possible to show that the sum of the total SSE and the total SSB is a constant; i.e., that it is equal to the total sum of squares (TSS), which is the sum of squares of the distance of each point to the overall mean of the data. The importance of this result is that minimizing SSE (cohesion) is equivalent to maximizing SSB (separation).

$$
\begin{aligned}
\text{TSS} &= \sum_{i=1}^{K} \sum_{x \in C_i} (x - c)^2 \\
&= \sum_{i=1}^{K} \sum_{x \in C_i} ((x - c_i) - (c - c_i))^2 \\
&= \sum_{i=1}^{K} \sum_{x \in C_i} (x - c_i)^2 - 2 \sum_{i=1}^{K} \sum_{x \in C_i} (x - c_i)(c - c_i) + \sum_{i=1}^{K} \sum_{x \in C_i} (c - c_i)^2 \\
&= \sum_{i=1}^{K} \sum_{x \in C_i} (x - c_i)^2 + \sum_{i=1}^{K} \sum_{x \in C_i} (c - c_i)^2 \\
&= \sum_{i=1}^{K} \sum_{x \in C_i} (x - c_i)^2 + \sum_{i=1}^{K} |C_i|(c - c_i)^2 \\
&= \text{SSE} + \text{SSB}
\end{aligned}
$$

The Silhouette Coefficient

The popular method of silhouette coefficients combines both cohesion and separation. The following steps explain how to compute the silhouette coefficient for an individual point, a process that consists of the **following three steps.** We use distances, but an analogous approach can be used for similarities.

1. For the i^{th} object, calculate its average distance to all other objects in its cluster. Call this value a_i.
2. For the i^{th} object and any cluster not containing the object, calculate the object's average distance to all the objects in the given cluster. Find the minimum such value with respect to all clusters; call this value b_i
3. For the i^{th} object, the silhouette coefficient is $s_i = (b_i - a_i)/\max(a_i, b_i)$

The value of the silhouette coefficient can vary between -1 and 1.

A negative value is undesirable because this corresponds to a case in which a the average distance to points in the cluster, is greater than ba, the minimum average distance to points in another cluster. We want the silhouette coefficient to be positive (ai <bi) and for a_i to be as close to 0 as possible, since the coefficient assumes its maximum value of 1 when at $a_i = 0$.

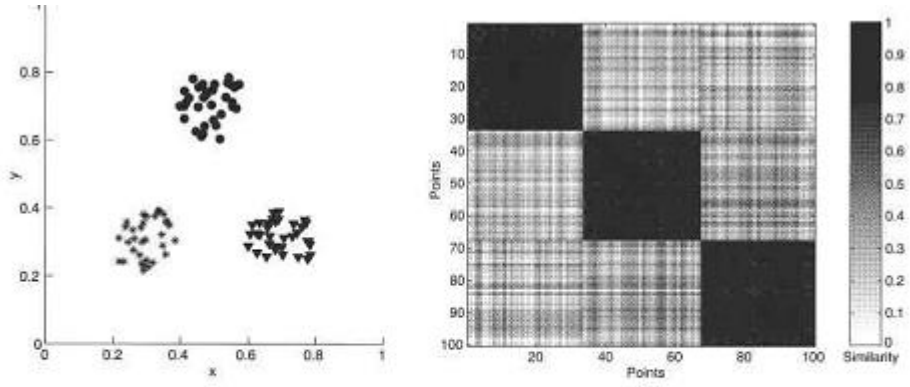
8.5.3 Unsupervised Cluster Evaluation Using the Proximity Matrix

Measuring Cluster Validity via Correlation

- If we are **given the similarity matrix** for a data set and the cluster labels from a cluster analysis of the data set, then **we can evaluate the "goodness"** of the clustering by looking at the correlation **between the similarity matrix and an ideal version of the similarity matrix** based on the cluster labels.

- More specifically, an ideal cluster is one whose points have a similarity of 1 to all points in the cluster, and a similarity of 0 to all points in other clusters.

- Thus the similarity is non-zero, i.e., 1, inside the blocks of the similarity matrix whose entries represent intra-cluster similarity, and 0 elsewhere.
- The ideal similarity matrix is constructed by creating a matrix that has one row and one column for each data point just like an actual similarity matrix and assigning a 1 to an entry if the associated pair of points belongs to the same cluster. All other entries are 0.
- High correlation between the ideal and actual similarity matrices indicates that the points that belong to the same cluster are close to each other, while low correlation indicates the opposite.
- Consequently, this is not a good measure for many density- or contiguity-based clusters, because they are not globular and may be closely intertwined with other clusters.



(a) Well-separated clusters.

(b) Similarity matrix sorted by K-means cluster labels.

Figure 8.30. Similarity matrix for well-separated clusters.

Example 8.10 (Visualizing a Similarity Matrix). Consider the points in Figure 8.30(a), which form three well-separated clusters. If we use K-means to group these points into three clusters, then we should have no trouble finding these clusters since they are well-separated. The separation of these clusters is illustrated by the reordered similarity matrix shown in Figure 8.30(b).

8.5.4 Unsupervised Evaluation of Hierarchical Clustering

A popular evaluation measure for hierarchical clusterings. The cophenetic distance between two objects is the proximity at which an agglomerative hierarchical clustering technique puts the objects in the same cluster for the first time.

For example, if at some point in the agglomerative hierarchical clustering process, the smallest distance between the two clusters that are merged is 0.1, then all points in one cluster have a cophenetic clistance of 0.1 with respect to the points in the other cluster. In a cophenetic distance matrix, the entries are the cophenetic distances between each pair of objects. The cophenetic distance is different for each hierarchical clustering of a set of points.

CoPhenetic Correlation Coefficient (CPCC) is the correlation between the entries of this matrix and the original dissimilarity matrix and is a standard measureof how well a hierarchical clustering (of a particular type) fits the data. One of the most common uses of this measure is to evaluate which type of hierarchical

clustering is best for a particular type of data.

Table 8.8. Cophenetic correlation coefficient for data of Table 8.3 and four agglomerative hierarchical clustering techniques.

Technique	CPCC
Single Link	0.44
Complete Link	0.63
Group Average	0.66
Ward's	0.64

8.5.6 Supervised Measures of Cluster Validity

we will refer to these two types of measures as classification-oriented and similarity-oriented, respectively.

8.5.6.1 Classification-Oriented Measures of Cluster Validity

There are a number of measures entropy, purity, precision, recall, and the F-measure that are commonly used to evaluate the performance of a classification model.

Entropy: The degree to which each cluster consists of objects of a single class.

- For each cluster, the class distribution of the data is calculated first, i.e., for cluster j we compute p_{ij}, the probability that a member of cluster i belongs to class j .
- we compute as Pij=mij/mi where mi is the number of objects in cluster i, and mii is the number of objects of class j in cluster i.
- Using this class distribution, the entropy of each cluster i is calculated using the standard formula

$$ei = -\sum_{j=1}^{L} Pij \quad \log_2 Pij$$

where L is the number of classes. The total entropy for a set of clusters is calculated as the sum of the entropies of each cluster weighted by the size of each cluster, i.e.,

$e = \sum_{i=1}^{K} \frac{mi}{m} ei$ where K is the number of clusters and rn is the total number of data points.

Purity: Another measure of ttLe extent to which a cluster contains objects of a single class. The purity of cluster i is pi = $^{max}_{j}\, pij$

The overall purity of a clustering is purity=$\sum_{i=1}^{K} \frac{mi}{m} Pi$

Precision: The fraction of a cluster that consists of objects of a specified class. The precision of cluster i with respect to class j is precision(i, j) =pij

Recall: The extent to which a cluster contains all objects of a specified class. The recall of cluster i with respect to class j is recall(i., j) := mij/ mj, where mj is the number of objects in class j.

F-measure : A combination of both precision and recall that measures the extent to which a cluster contains only objects of a particular class and all objects of that class. The F-measure of cluster i with respect to class j is F(i,, j): (2x precision(*i*, j) x recall(i, j))/(precision(i,, j)*recall(i, j)).

8.5.6.2 Similarity-Oriented Measures of Cluster Validity

The measures that are all based on the premise that any two objects that are in the same cluster should be in the same class and vice versa. We can view this approach to cluster validity as involving the comparison of two matrices:

(1) the ideal cluster similarity rnatrix discussed previously, which has a 1 in the i,jth entry if two objects, i. and j,are in the same cluster and 0, otherwise, and

(2) an ideal class similarity matrix defined with respect to class labels, which has a 1 in the i,jth entry if two objects, i and j, belong to the same class, and a 0 otherwise. As before, we can take the correlation of these two matrices as the measure of cluster validity. This measure is known as the Γ statistic in clustering validation literature.

Example 8.16 (Correlation between Cluster and Class Matrices). To demonstrate this idea more concretely, we give an example involving five data points, p1,p2,p3,p4,p5, two clusters, C1: {p1,p2,p3} and C2: {p4,p5}, and two classes, L1 : {p1,,p2} and L2 : {p3,p4,p5}.

The ideal cluster and class similarity matrices are given in Tables 8.10 and 8.11. The correlation between the entries of these two matrices is 0.359.

Table 8.10. Ideal cluster similarity matrix.

Point	p1	p2	p3	p4	p5
p1	1	1	1	0	0
p2	1	1	1	0	0
p3	1	1	1	0	0
p4	0	0	0	1	1
p5	0	0	0	1	1

Table 8.11. Ideal class similarity matrix.

Point	p1	p2	p3	p4	p5
p1	1	1	0	0	0
p2	1	1	0	0	0
p3	0	0	1	1	1
p4	0	0	1	1	1
p5	0	0	1	1	1

1.Density-Based Clustering

1.Grid based clustering :

The idea is to split the possible values of each attribute into a number of contiguous intervals, creating a set of grid cells.

Each object falls into a grid cell whose corresponding attribute intervals contain the values of the object.

Algorithm 9.4 Basic grid-based clustering algorithm.

1: Define a set of grid cells.
2: Assign objects to the appropriate cells and compute the density of each cell.
3: Eliminate cells having a density below a specified threshold, τ.
4: Form clusters from contiguous (adjacent) groups of dense cells.

The Density of Grid Cells

(Grid-Based Density). Figure 9.10 shows two sets of two dimensional points divided into 49 cells using a 7-by-7 grid.

The first set contains 200 points generated from a uniform distribution over a circle centered

at (2, 3) of radius 2, while the second set has 100 points generated from a uniform distribution over a circle centered at (6, 3) of radius 1.

The counts for the grid cells are shown in Table 9.2. Since the cells have equal volume (area), we can consider these values to be the densities of the cells.

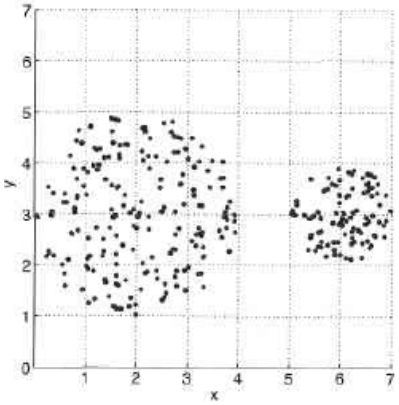
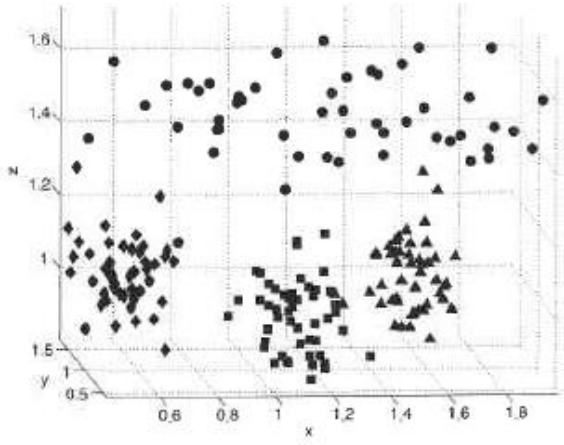


Figure 9.10. Grid-based density.

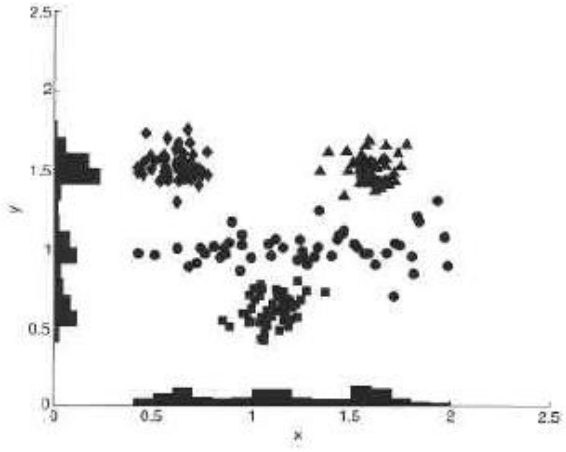
0	0	0	0	0	0	0
0	0	0	0	0	0	0
4	17	18	6	0	0	0
14	14	13	13	0	18	27
11	18	10	21	0	24	31
3	20	14	4	0	0	0
0	0	0	0	0	0	0

Table 9.2. Point counts for grid cells.

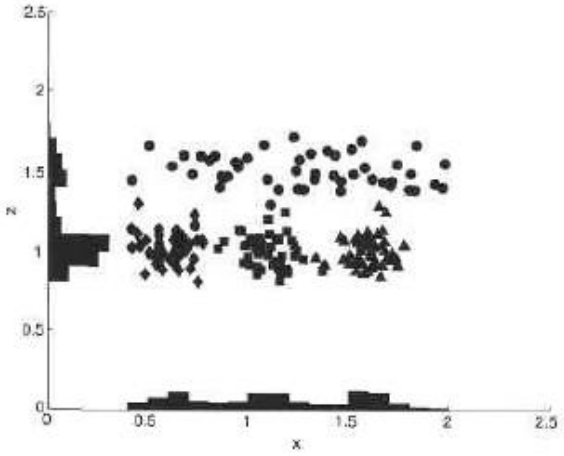
Subspace Clustering



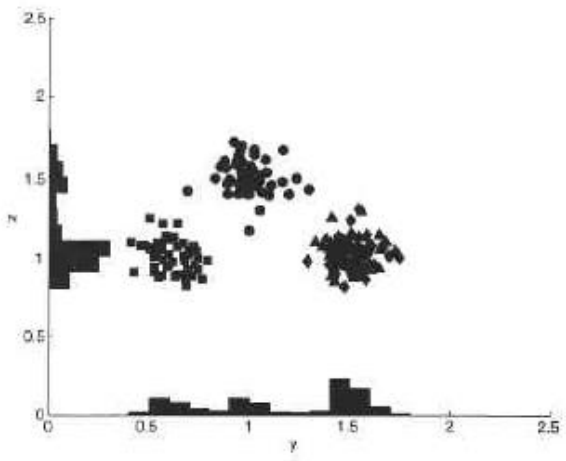
(a) Four clusters in three dimensions.



(b) View in the xy plane.



Yz plane



xz plane

Example 9.9 (Subspace Clusters). Figure 9.11(a) shows a set of points in three-dimensional space. There are three clusters of points in the full space, which are represented by squares) diamonds, and triangles. In addition, there is one set of points, represented by circles, that is not a cluster in three dimensional space.

CLIQUE

Consider a set of adjacent, k-dimensional cells that form a cluster; i.e., there is a collection of adjacent cells that have a density above the specified threshold (. A corresponding set of cells in k - 1 dimensions can be found by omitting one of the k dimensions (attributes). The lower-dimensional cells are still adjacent, and each low-dimensional cell contains all points of the corresponding high-dimensional cell. It may contain additional points as well. Thus, a low dimensional cell has a density greater than or equal to that of its corresponding high-dimensional cell.

Algorithm 9.5 CLIQUE.

1: Find all the dense areas in the one-dimensional spaces corresponding to each attribute. This is the set of dense one-dimensional cells.
2: $k \leftarrow 2$
3: **repeat**
4: Generate all candidate dense k-dimensional cells from dense $(k-1)$-dimensional cells.
5: Eliminate cells that have fewer than ξ points.
6: $k \leftarrow k+1$
7: **until** There are no candidate dense k-dimensional cells.
8: Find clusters by taking the union of all adjacent, high-density cells.
9: Summarize each cluster using a small set of inequalities that describe the attribute ranges of the cells in the cluster.

2. Graph-Based Clustering

The following are some key approaches, different subsets of which are employed by these algorithms.

1. Sparsify the proximity graph to keep only the connections of an object with its nearest neighbors. This sparsification is useful for handling noise and outliers. It also allows the use of highly efficient graph partitioning algorithms that have been developed for sparse graphs.

2. Define a similarity measure between two objects based on the number of nearest neighbors that they share. This approach, which is based on the observation that an object and its nearest neighbors usually belong to the same class, is useful for overcoming problems with high dimensionality and clusters of varying density.

Define core objects and build clusters around them. To do this for graph based clustering, it is necessary to introduce a notion ofdensity-based on a proximity graph or a sparsified proximity graph. As with DBSCAN, building clusters around core objects leads to a clustering technique that can find clusters of differing shapes and sizes.

Use the information in the proximity graph to provide a more sophisticated evaluation of whether two clusters should be merged. Specifically, two clusters are merged only if the resulting cluster will have characteristics similar to the original two clusters.

Sparsification

Sparsification has several beneficial effects: Data size is reduced. The amount of data that needs to be processed to cluster the data is drastically reduced. Sparsification can often eliminate more than 99% of the entries in a proximity matrix.

As a result, the size of problems that can be handled is increased. Clustering may work better. Sparsification techniques keep the connections to their nearest neighbors of an object while breaking the connections to more distant objects. This is in keeping with the nearest neighbor principle that the nearest neighbors of an object tend to belong to the same class (cluster) as the object itself.

This reduces the impact of noise and outliers and sharpens the distinction between clusters.

Graph partitioning algorithms can be used. There has been a considerable amount of work on heuristic algorithms for finding min-cut partitionings of sparse graphs, especially in the areas of parallel computing and the design of integrated circuits. Sparsification of the proximity graph makes it possible to use graph partitioning algorithms for the clustering process.

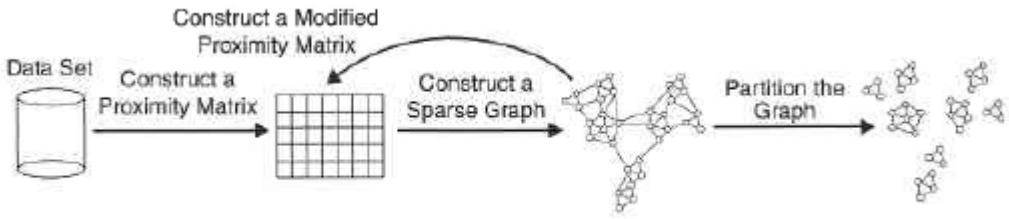


Figure 9.15. Ideal process of clustering using sparsification.

2. Minimum Spanning Tree (MST) Clustering

A minimum spanning tree of a graph is a subgraph that

(1) has no cycles, i.e., is a tree,

(2) contains all the nodes of the graph,

(3) has the minimum total edge weight of all possible spanning trees.

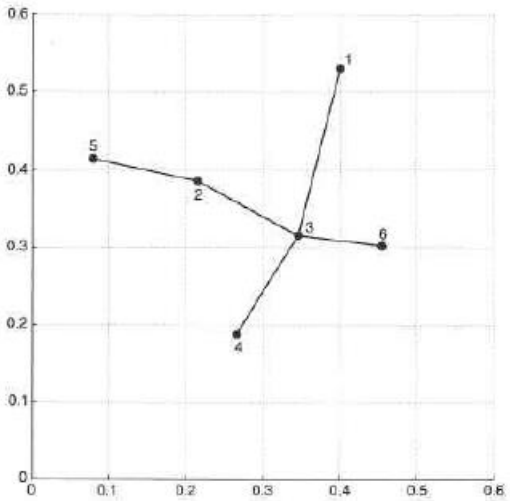


Figure 9.16. Minimum spanning tree for a set of six two-dimensional points.

Algorithm 9.7 MST divisive hierarchical clustering algorithm.

1: Compute a minimum spanning tree for the dissimilarity graph.
2: **repeat**
3: Create a new cluster by breaking the link corresponding to the largest dissimilarity.
4: **until** Only singleton clusters remain.

2. OPOSSUM: Optimal Partitioning of Sparse Similarities Using METIS

OPOSSUM is a clustering technique that was specifically designed for clustering sparse, high dimensional data, such as document or market basket data.

OPOSSUM uses the METIS algorithm, which was specifically created for partitioning sparce graphs.

Algorithm 9.8 OPOSSUM clustering algorithm.

1: Compute a sparsified similarity graph.
2: Partition the similarity graph into k distinct components (clusters) using METIS.

The METIS graph partitioning program partitions a sparse graph into k distinct components, where k is a user-specified parameter, in order to

(1) minimize the weight of the edges (the similarity) between components

(2) fulfil a balance constraint.

OPOSSUM uses one of the following two balance constraints:

(1) the number of objects in each cluster must be roughly the same,

(2) the sum of the attribute values must be roughly the same.

3.DENCLUE: A Kernel-Based Scheme for Density-Based Clustering

DENCLUE (DENsity ClUstEring) is a density-based clustering approach that models the overall density of a set of points as the sum of influence functions associated with each point.

The resulting overall density function will have local peaks, i.e., local density maxima, and these local peaks can be used to define clusters in a natural way.

For each data point, a hill climbing procedure finds the nearest peak associated with that point, and the set of all data points associated with a particular peak (called a local density attractor) becomes a cluster.

If the density at a local peak is too low, then the points in the associated cluster are classified as noise and discarded.

If a local peak can be connected to a second local peak by a path of data points, and the density at each point on the path is above the minimum density threshold, then the clusters associated with these local peaks are merged.

Algorithm 9.6 DENCLUE algorithm.

1: Derive a density function for the space occupied by the data points.
2: Identify the points that are local maxima.
(These are the density attractors.)
3: Associate each point with a density attractor by moving in the direction of maximum increase in density.
4: Define clusters consisting of points associated with a particular density attractor.
5: Discard clusters whose density attractor has a density less than a user-specified threshold of ξ.
6: Combine clusters that are connected by a path of points that all have a density of ξ or higher.

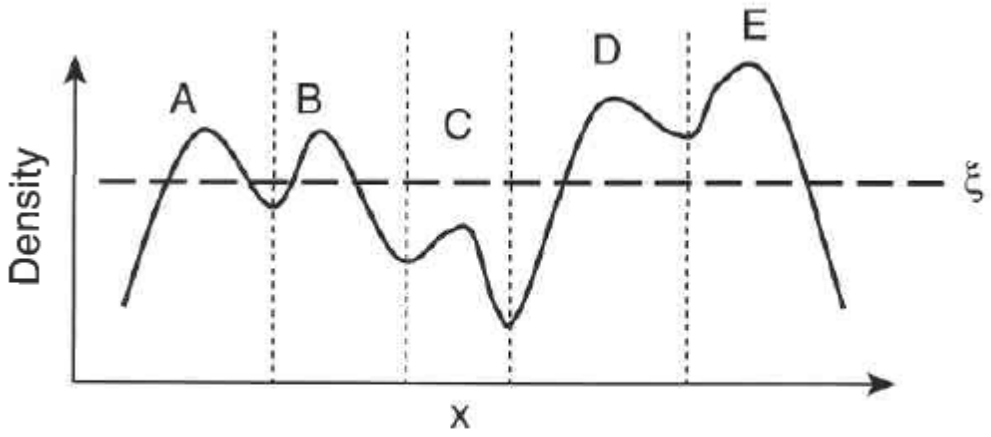


Figure 9.13. Illustration of DENCLUE density concepts in one dimension.

4. Chameleon: Hierarchical Clustering with Dynamic Modeling

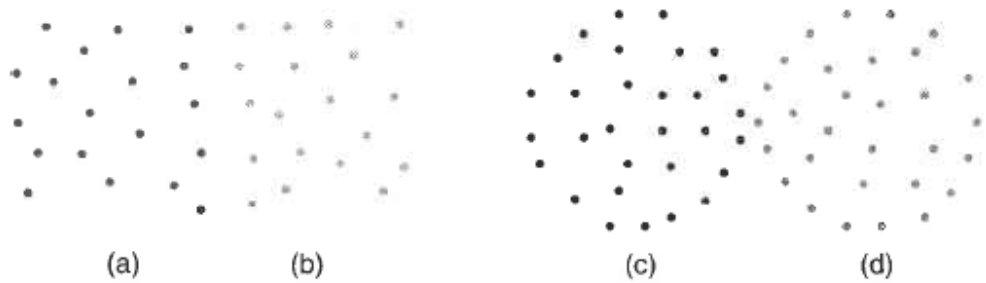


Figure 9.17. Situation in which closeness is not the appropriate merging criterion. ©1999, IEEE

Figure shows four clusters.

If we use the closeness of clusters (as measured by the closest two points in different clusters) as our merging criterion, then we would merge the two circular clusters, (c) and (d),which almost touch, instead of the rectangular clusters, (a) and (b), which are separated by a small gap.

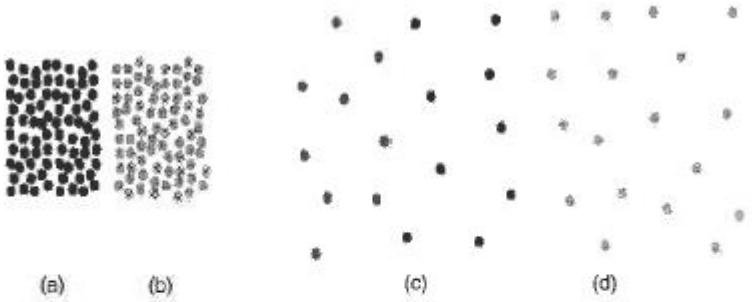


Figure 9.18. Illustration of the notion of relative closeness. ©1999, IEEE

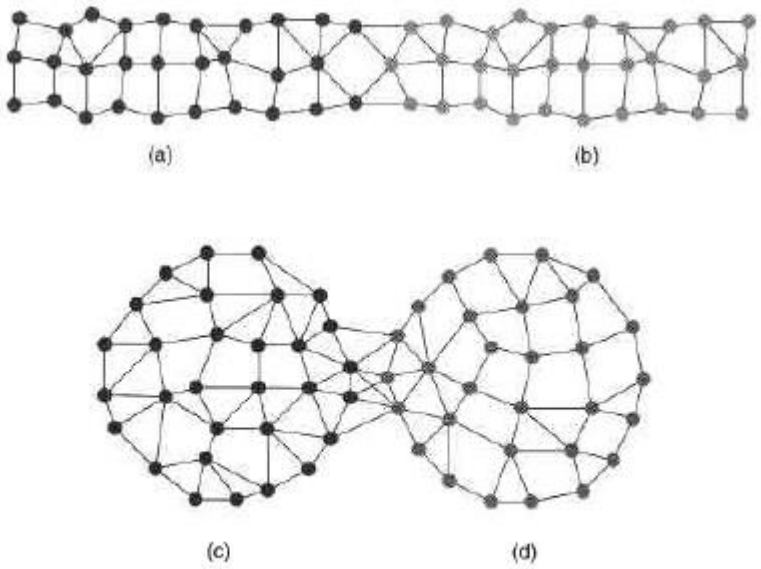


Figure 9.19. Illustration of the notion of relative interconnectedness. ©1999, IEEE

(a) and (b) are closer in absolute terms than clusters (c) and (d), this is not true if the nature of the clusters is taken into account.

Deciding Which Clusters to Merge :

Relative Closeness (RC) is the absolute closenesso f two clusters normalized bv the internal closenesso f the clusters. Two clusters are combined only if the points in the resulting cluster are almost as close to each other as in each of the original clusters. Mathematically,

$$RC = \frac{\bar{S}_{EC}(C_i, C_j)}{\frac{m_i}{m_i+m_j}\bar{S}_{EC}(C_i) + \frac{m_j}{m_i+m_j}\bar{S}_{EC}(C_j)},$$

where rni and mj are the sizes of clusters Ci and Cj, respectively, Sec(Ci,Cj) is the average weight of the edges (of the k-nearest neighbour graph) that connect clusters Ci and Cj; Sec(Ci) is the average weight of edges if we bisect cluster Ci and Sec(Cj) is the average weight of edges if we bisect cluster Cj. (EC stands for edge cut.) Figure 9.18 illustrates the notion of relative closeness.

Relative Interconnectivity (RI) is the absolute interconnectivity of two clusters normalized by the internal connectivity of the clusters. Two clusters are combined if the points in the resulting cluster are almost as strongly connected as points in each of the original clusters. Mathematicallv.

$$RI = \frac{EC(C_i, C_j)}{\frac{1}{2}(EC(C_i) + EC(C_j))},$$

where EC(Ci,Cj) is the sum of the edges (of the k-nearest neighbour graph) that connect clusters Ci and Cj; EC(Cj) is the minimum sum of the cut edges if we bisect cluster Ci; and EC(Cj) is the minimum sum of the cut edges if we bisect cluster Cj. Figure 9.19 illustrates the notion of relative interconnectivity.

The two circular clusters, (c) and (d), have more connections than the rectangular clusters, (a) and (b).

However, merging (c) and (d) produces a cluster that has connectivity quite different from that of (c) and (d). In contrast, merging (a) and (b) produces a cluster with connectivity very similar to that of (a) and (b).

RI and RC can be combined in many different ways to yield an overall measure of self-similarity.

Approach used in Chameleon is to merge the pair of clusters that maximizes RI(Ci,Cj) * RC(Ci,Cj), where a is a user specified parameter that is typically greater than 1.

Chameleon Algorithm

Chameleon consists of three key steps: sparsification, graph partitioning, and hierarchical clustering. Algorithm 9.9 and Figure 9.20 describe these steps.

Algorithm 9.9 Chameleon algorithm.

1: Build a k-nearest neighbor graph.
2: Partition the graph using a multilevel graph partitioning algorithm.
3: **repeat**
4: Merge the clusters that best preserve the cluster self-similarity with respect to relative interconnectivity and relative closeness.
5: **until** No more clusters can be merged.

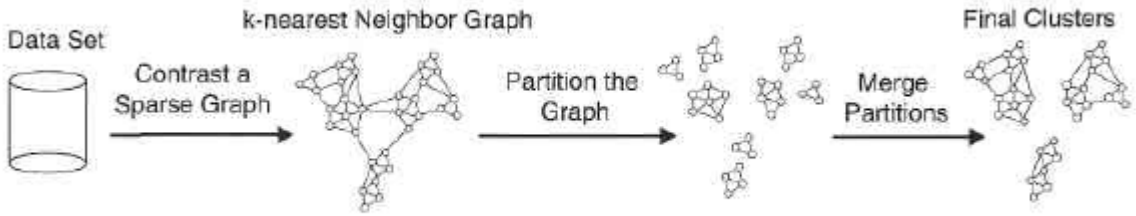


Figure 9.20. Overall process by which Chameleon performs clustering. ©1999, IEEE

Sparsification The first step in Chameleon is to generate a k-nearest neighbour graph. Conceptually, such a graph is derived from the proximity graph, and it contains links only between a point and its k nearest neighbors, i.e., the points to which it is closest.

Graph Partitioning

Once a sparsified graph has been obtained, an efficient multilevel graph partitioning algorithm, such as METIS (see bibliographic notes) can be used to partition the data set. Chameleon starts with an all-inclusive graph (cluster) and then bisects the largest current subgraph (cluster) until no cluster has more than MIN-SIZE points, where MIN_SIZE is a user-specified parameter.

This process results in a large number of roughly equally sized groups of wellconnected vertices (highly similar data points).

Agglomerative Hierarchical Clustering

Chameleon merges clusters based on the notion of self-similarity. Chameleon can be parameterized to merge more than one pair of clusters in a single step and to stop before all objects have been merged into a single cluster.

Complexity Assume that rn is the number of data points and p is the number of partitions. For lowdimensional data, this takes O(mlogrn). for high-dimensional data sets, the time complexity of the sparsification becomes $O(m^2)$.

Example : Chameleon was applied to two data sets that clustering algorithms such as K-means and DBSCAN have difficulty clustering. The results of this clustering are shown in Figure 9.21. The clusters are identified by the shading of the points. In Figure 9.21,(a),t he two clusters are irregularly shaped and quite close to each other. AIso, noise is present. In Figure 9.21(b), the two clusters are connected by a bridge, and again, noise is present.

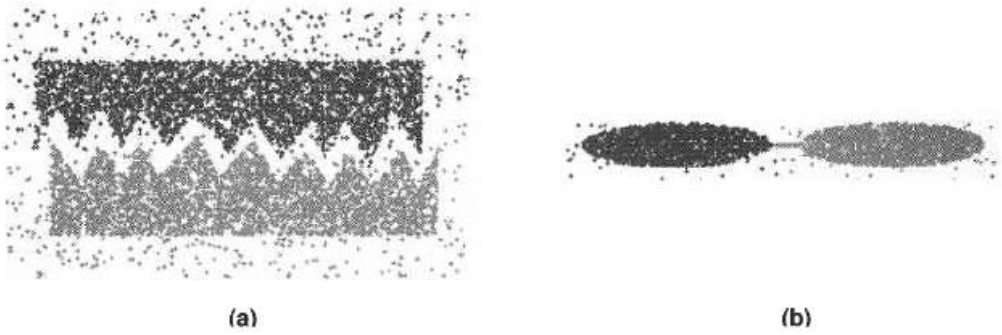


Figure 9.21. Chameleon applied to cluster a pair of two-dimensional sets of points. ©1999, IEEE

Shared Nearest Neighbor Similarity :

If two points are similar to many of the same points, then they are similar to one another, even if a direct measurement of similarity does not indicate this.

SNN Similarity Computation

SNN similarity is the number of shared neighbors as long as the two objects are on each other's nearest neighbor lists.

Algorithm 9.10 Computing shared nearest neighbor similarity

1: Find the k-nearest neighbors of all points.
2: **if** two points, $\mathbf{x}$ and $\mathbf{y}$ are *not* among the k-nearest neighbors of each other **then**
3: $\quad similarity(\mathbf{x}, \mathbf{y}) \leftarrow 0$
4: **else**
5: $\quad similarity(\mathbf{x}, \mathbf{y}) \leftarrow$ number of shared neighbors
6: **end if**

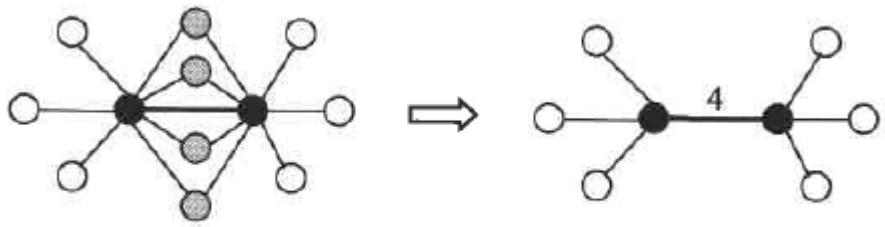


Figure 9.23. Computation of SNN similarity between two points.

The computation of SNN similarity is described by Algorithm 9.10 and graphically illustrated by Figure 9.23. Each of the two black points has eight nearest neighbors, including each other. Four of those nearest neighbours the points in gray are shared. Thus, the shared nearest neighbor similarity between the two points is 4.

The similarity graph of the SNN similarities among objects is called the SNN similarity graph. Since many pairs of objects will have an SNN similarity of 0, this is a very sparse graph.

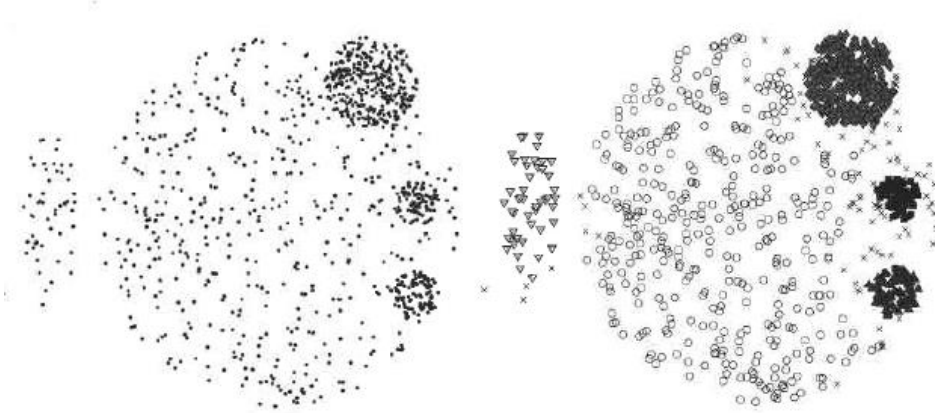
The Jarvis-Patrick Clustering Algorithm

The JP clustering algorithm replaces the proximity between two points with the SNN similarity, which is calculated as described in Algorithm 9.10. A threshold is then used to sparsify this matrix of SNN similarities. In graph terms, an SNN similarity graph is created and sparsified.

Clusters are simply the connected components of the SNN graph.

Algorithm 9.11 Jarvis-Patrick clustering algorithm.

1: Compute the SNN similarity graph.
2: Sparsify the SNN similarity graph by applying a similarity threshold.
3: Find the connected components (clusters) of the sparsified SNN similarity graph.



(a) Original data. (b) Clusters found by Jarvis-Patrick.

Figure 9.24. Jarvis-Patrick clustering of a two-dimensional point set.

Eg. (JP Clustering of a Two-Dimensional Data Set).

JP clustering is applied to the "fish" data set shown in Figure 9.2a@) to find the clusters shown in Figure 9.24(b). The size of the nearest neighbor list was 20, and two points were placed in the same cluster if they shared at least 10 points. The different clusters are shown by the different markers and different shading. The points whose marker is an "x" were classified as noise by Jarvis- Patrick.

SNN Density

SNN density measures the degree to which a point is surrounded by similar points (with respect to nearest neighbors). Thus, points in regions of high and low density will typically have relatively high SNN density, while points in regions where there is a transition from low to high density-points that are between clusters will tend to have low SNN density

(a) All points. (b) High SNN density. (c) Medium SNN density. (d) Low SNN density.

Figure 9.25. SNN density of two-dimensional points.

Core points. A point is a core point if the number of points within a given neighborhood around the point, as determined by SNN similarity and a supplied parameter -Eps exceeds a certain threshold MinPts, which is also a supplied parameter.

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

Noise points. A noise point is any point that is neither a core point nor a border point.

The SNN Density-based Clustering Algorithm

The steps of the SNN density-based clustering algorithm are shown in AIgorithm

9.12.

Algorithm 9.12 SNN density-based clustering algorithm.

1: Compute the SNN similarity graph.
2: Apply DBSCAN with user-specified parameters for *Eps* and *MinPts*.

Scalable Clustering Algorithms :

1. Scalability: General Issues and Approaches
2. Multidimensional or Spatial Access Methods
3. Bounds on Proximities
4. Sampling
5. Partitioning the data objects
6. Summarization
7. Parallel and Distributed Computation

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) :

Algorithm 9.13 BIRCH.

1: **Load the data into memory by creating a CF tree that summarizes the data.**
2: **Build a smaller CF tree if it is necessary for phase 3.** T is increased, and then the leaf node entries (clusters) are reinserted. Since T has increased, some clusters will be merged.
3: **Perform global clustering.** Different forms of global clustering (clustering that uses the pairwise distances between all the clusters) can be used. However, an agglomerative, hierarchical technique was selected. Because the clustering features store summary information that is important to certain kinds of clustering, the global clustering algorithm can be applied as if it were being applied to all the points in a cluster represented by the CF.
4: **Redistribute the data points using the centroids of clusters discovered in step 3, and thus, discover a new set of clusters.** This overcomes certain problems that can occur in the first phase of BIRCH. Because of page size constraints and the T parameter, points that should be in one cluster are sometimes split, and points that should be in different clusters are sometimes combined. Also, if the data set contains duplicate points, these points can sometimes be clustered differently, depending on the order in which they are encountered. By repeating this phase multiple times, the process converges to a locally optimum solution.

- BIRCH is based on the notion of a Clustering Feature (CF) and a CF tree.
- A cluster of data points (vectors) can be represented by a triple of numbers (N, LS, SS), where N is the number of points in the cluster, LS is the linear sum of the points and SS is the sum of squares of the points.

- Compute the centroid of a cluster and its variance (standard deviation) to measure diameter & distance between clusters.

- A CF tree is a height-balanced tree. Each interior node has entries of the form [CF, $child_i$], where $child_i$ is a pointer to the ith child node.

- Leaf nodes are subject to the restriction that each leaf node must have a diameter that is less than a parameterized threshold.

- By adjusting the threshold parameter T, the height of the tree can be controlled. T controls the fineness of the clustering, i.e., the extent to which the data in the original set of data is reduced.

- As each data point is encountered, 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 point is finally identified.

- A test is performed to see if adding the data item to the candidate cluster will result in a new cluster with a diameter greater than the given threshold.

- If not, then the data point is added to the candidate cluster.

- If the new cluster has a diameter greater than T, then a new entry is created if the leaf node is not full. Otherwise the leaf node must be split.

- The two entries (clusters) that are farthest apart are selected as seeds and the remaining entries are distributed to one of the two new leaf nodes, based on which leaf node contains the closest seed cluster.

- Once the leaf node has been split, the parent node is updated and split if necessary; i.e., if the parent node is full. This process may continue all the way to the root node.

  BIRCH follows each split with a merge step.

- At the interior node where the split stops, the two closest entries are found. If these entries do not correspond to the two entries that just resulted from the split, then an attempt is made to
- merge these entries and their corresponding child nodes.

- BIRCH also has a procedure for removing outliers.

CURE (Clustering Using Representatives):

CURE represents a cluster by using multiple representative points from the cluster.

The first representative point is chosen to be the point farthest from the center of the cluster, while the remaining points are chosen so that they are farthest from all the previously chosen points.

The number of points chosen
is a parameter, but it was found that a value of 10 or more worked well.

Once the representative points are chosen, they are shrunk toward the center by a factor, a.

For example, a representative point that was a distance of 10 units from the center would move by 3 units (for a =0.7), while a representative point at a distance of 1 unit would only move 0.3 units.

K is the desired number of clusters, m is the number of points, p is the number of partitions, and q is the desired reduction of points in a partition.

The number of clusters in a partition is m/pq.

Therefore, the total number of clusters is m/pq.

For example, if m = 10,000, p = 10 and q = 100, then each partition contains 10,000/10 = 1000 points, and there would be 1000/100 = 10 clusters in each partition and 10,000/100 = 100 clusters overall.

Algorithm 9.14 CURE.

1: **Draw a random sample from the data set**. The CURE paper is notable for explicitly deriving a formula for what the size of this sample should be in order to guarantee, with high probability, that all clusters are represented by a minimum number of points.
2: **Partition the sample into p equal-sized partitions.**
3: **Cluster the points in each partition into $\frac{m}{pq}$ clusters using CURE's hierarchical clustering algorithm to obtain a total of $\frac{m}{q}$ clusters.** Note that some outlier elimination occurs during this process.
4: **Use CURE's hierarchical clustering algorithm to cluster the $\frac{m}{q}$ clusters found in the previous step until only K clusters remain.**
5: **Eliminate outliers.** This is the second phase of outlier elimination.
6: **Assign all remaining data points to the nearest cluster to obtain a complete clustering.**