A Synthetic Data Generator for Clustering and Outlier Analysis

Yaling Pei, Osmar Zaïane

Computing Science Department
University of Alberta, Edmonton, Canada T6G 2E8
{yaling, zaiane}@ualberta.ca

Abstract. We present a distribution-based and transformation-based approach to synthetic data generation and demonstrate that the approach is very efficient in generating different types of multi-dimensional numerical datasets for data clustering and outlier analysis. We developed a data generating system that is able to systematically create testing datasets based on user's requirements such as the number of points, the number of clusters, the size, shapes and locations of clusters, and the density level of either cluster data or noise/outliers in a dataset. Two standard probability distributions are considered in data generation. One is uniform distribution and the other is normal distribution. Since outlier detection, especially local outlier detection, is conducted in the context of clusters of a dataset, our synthetic data generator is suitable for both clustering and outlier analysis. In addition, the data format has been carefully designed so that generated data can be visualized not only by our system but also by some popular statistical rendering tools such as statCrunch [16] and statPoint [17] that display data with standard statistical graphical approaches. To our knowledge, our system is probably the first synthetic data generation system that systematically generates datasets for evaluating the clustering and outlier analysis algorithms. Being an object-oriented system, the current data generator can be easily integrated into other data analysis systems.

1 Introduction

Clustering analysis and outlier detection are two important techniques widely used in data mining and automatic knowledge discovery. Although research on outlier analysis is relatively new in the area of data mining compared to data clustering, they have both been addressed by many researchers and there exist a large number of approaches to clustering and outlier analysis. While different algorithms have their own strength in finding clusters and/or outliers, the performance of a particular algorithm can be quite different with different datasets. Therefore, the choice of clustering or outlier analysis methods depends on the specific purpose of the application as well as the datasets available. This in turn poses one of the most important issues in data analysis: How do we assess a data analysis algorithm?

It is hard to say that one algorithm is better than the other since different algorithms usually use different testing datasets with certain constraints such as data distribution, dimension and density in the analysis of the effectiveness and efficiency. There exist some databases with a variety of datasets obtained from real life environment. These datasets could be in various formats and distributions that make it difficult to be used in testing and comparing different clustering and/or outlier algorithms. Surprisingly, little work has been done on systematically generating artificial datasets for the analysis and evaluation of data analysis algorithms in data mining area.

In this work, we explore the idea to automatically generate datasets in two or more dimensional space given the total number of points N and the number of clusters K in a dataset. We use data points to represent objects with multiple attributes. The properties of each dataset, including space between clusters, cluster distributions and outlier management are specified by the user but controlled automatically by the system. Each dataset is generated along with a difficulty level, a density level, an outlier level and a certain data distribution. Given a fixed number of points in a dataset, the size and density of clusters are closely related and are both controlled by the density level. The spreading and density of outliers with respect to the main body of the data is determined by the outlier level. The difficulty level is defined in terms of the existing clustering algorithms and they are roughly classified into three groups:

- easy level - the datasets at this level have only spherical or convex clusters;
- medium level- the datasets have long thin or arbitrarily shaped clusters;
- difficult level - the datasets can have clusters within clusters with all possible shapes.

The data generator can be used not only in the evaluation and testing of data clustering analysis and outlier detection but also in visualizing various data distributions . Our goal is to develop a generic framework for the generation of testing datasets with controlled level of clustering difficulties and devise a heuristic that will be improved upon in a meaningful way in high-dimensional and categorical space in the future. We investigate current research and implementation on data generation and proceed in different stages. An important part of data generation is to display the produced datasets in a graphical user interface for visual inspection. Hence, we combine the algorithm design with the implementation together in each stage of the development. Several methods such as distribution-based approaches and transformation-based approaches, or their combination have been employed in generating meaningful datasets. Java Swing is used as the programming language as the implementation of the data generation system relies heavily on the graphical user interface (GUI).

2 Existing Work on Synthetic Data Generation

An important issue in evaluating data analysis algorithms is the availability of representative data. When real-life data are hard to obtain or when their properties are hard to modify for testing various algorithms, synthetic data becomes

an appealing alternative. Most existing work on clustering and outlier analysis uses both synthetic data and real-life data to test the validity and performance of the proposed algorithms.

Data generation has been an important topic in mathematics and statistics. There are some state-of-the-art techniques on generating data of certain distribution, for example, random sequences and normal distribution, which serve as the fundamental tools for synthetic data generation systems in many applications. Despite increasing interest, the research on synthetic data generation in the area of data mining is still in its early stage. There exist some well-known datasets that have been widely used as benchmark datasets to test the performance of many clustering algorithms. Among them, one is provided by the team that developed the clustering algorithm CHAMELEON [9]. The dataset has 10,000 2D points and includes not only different shapes of clusters but also different type of outliers. Unfortunately, there is no description of how these datasets are generated.

In the literature of software testing, a large number of methods to automate test data generation have been studied [4]. In recent years, research areas such as data mining [8], sensor networks [21], artificial intelligence [14] and bioinformatics [19] are paying more attention in developing data generation systems aiming to systematically generate synthetic data for numerous applications. In this chapter, we will briefly discuss some existing data generation methods and systems.

2.1 IBM Quest Synthetic Data Generator

A well-known synthetic data generation system is developed by the IBM's QUEST data mining group [8]. The system consists of two data generators. One is used to generate transaction data for mining associations and sequential patterns. Given some parameters, the system can produce a set of data containing information of customer transactions. The other generator produces data intended for the task of classification. The output is a person database in which each entry has nine attributes. QUEST also developed a series of classification functions of increasing complexity that used the nine attributes to classify people into different groups.

The generated datasets contain only numerical values. Values of non-numerical attributes are converted to numerical values according to some pre-defined rules.

2.2 Synthetic Data Generation in Other Research Fields

Synthetic data generation also plays an important rule in many different fields of computer science such as Information retrieval, software engineering and artificial intelligence, although in each field the focus and the requirement of generating synthetic data are quite different.

The GSTD algorithm proposed in [18] uses three operations to generate spatiotemporal datasets by gradually altering the three parameters that control the

duration, the location, and the size of spatiotemporal objects. Such a data generator serves as an integral part of the benchmark environment for spatiotemporal data access system.

The main focus of test data generation in automatic software testing is to generate input data to test the correctness of a given computer program or software system. To have a sufficient coverage on the execution of a computer program, a data generation system first needs to analyze the control flow of the program to identify target execution paths to be tested. Input data with which the execution of the program follows a specific path are usually generated by using either symbolic evaluation techniques or solving a properly formulated optimization problem.

In the field of artificial intelligence, many important problems are NP-hard such as the Boolean satisfiability problem (SAT). To test the performance of solvers and algorithms for these problems, one also needs to generate testing problem instances.

In addition to real-world and manually compiled benchmarks, a recent trend is to generate problem instances randomly from some probability distribution. As a matter of fact, the study of the typical-case hardness of randomly-generated problem instances and the performance of various algorithms on these instances has been an important research topic in artificial intelligence. On the one hand, many deep theoretical results on the complexity of NP-hard problems and useful insights into the design of more efficient algorithm have been obtained. On the other hand, hard testing problem instances generated at the so-called phase transition region of some random problem model have been one of the driving forces in the development of the start-of-the-art solvers for these AI problems.

3 Mathematical Tools and Techniques

In an effort to systematically generate test datasets for data analysis, we make use of some mathematical tools such as probability distributions and linear transformations. By applying these principles, the proposed method provides the mechanism that datasets are not only generated automatically but also controlled by the parameters from the user input. This section introduces the mathematical concepts and tools related to our proposed approach.

3.1 Uniform Distribution

The uniform distribution is the simplest continuous distribution in probability. A random variable x has the uniform distribution if all possible values of the variable are equally probable [13]. It is also called rectangular distribution.

Uniform distribution is specified by two parameters: the end points a and b. The distribution has constant probability density on the interval (a, b) and zero probability density elsewhere. The probability density function(PDF) and cumulative distribution function(CDF) for a continuous uniform distribution on

(a, b) are

$$f(x) = \begin{cases} \frac{1}{b-a}, & a < x < b; \\ 0, & \text{otherwise.} \end{cases}$$

$$F(x) = \begin{cases} 0, & x < a; \\ \frac{x-a}{b-a}, & a < x < b; \\ 1, & x > b. \end{cases}$$

Fig. 1. PDF and CDF of uniform distribution

In Figure 1, (a) is the plot of the uniform PDF and (b) is the plot of the uniform CDF. Standard uniform distribution is the case where $a = 0$ and $b = 1$.

We aim to generate data from a multivariate uniform distribution. The dataset D is composed of a set of multi-dimensional points. Each point $(x_1, x_2, ..., x_m)$ is obtained by generating uniform random numbers for x_i, where $i = 1, 2, ..., m$. The values of each variable are uniformly distributed in $(0, 1)$. Since the joint distribution of two or more independent one-dimensional uniform distributions is also uniform, the points in D are uniformly distributed on a region in the feature space of all variables.

3.2 Normal Distribution

A continuous random variable x has a normal distribution or Gaussian distribution if its probability density function is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2},$$

where μ is mean, σ^2 is the variance and $-\infty < x < \infty$ [13].

Figure 2(a) is the plot of the normal PDF and Figure 2(b) is the plot of the normal CDF respectively. Standard normal distribution is the normal distribution given $\mu = 0$ and $\sigma^2 = 1$.

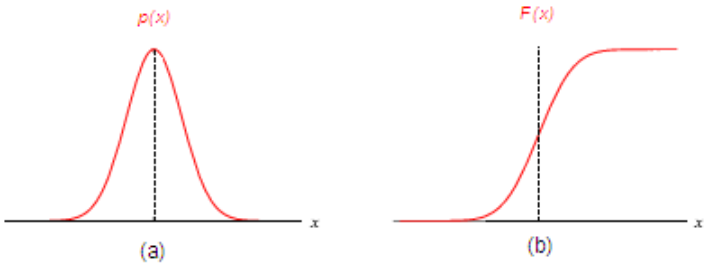


Fig. 2. PDF and CDF of normal distribution

Recall that the joint distribution of two independent one-dimensional normal distributions is a bivariate normal distribution. We can therefore generate normally distributed data points (x, y) in the plane by generating x and y independently from the one-dimensional standard normal distribution. The plot of a dataset of bivariate normal distribution has a round shape with the center being the origin of the coordinates and the inner points having a higher density than that of the outer points. Such method can be generalized to higher dimensions to generate normally distributed data with more attributes.

3.3 Box-muller Transformation

Box-Muller transformation allows us to transform a two-dimensional continuous uniform distribution to a two-dimensional bivariate normal distribution (or complex normal distribution) [12]. Let x_1 and x_2 be two independent random variables and are uniformly distributed between 0 and 1. The basic form of Box-Muller transformation is defined as

$$y_1 = \sqrt{-2\ln x_1} cos(2\pi x_2),$$
$$y_2 = \sqrt{-2\ln x_1} sin(2\pi x_2),$$

where y_1 and y_2 have a normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$.

In our data generation system, rather than using the normal probability density function to generate normal distribution, which costs too much in computation, we adopted Box-muller transformation. By applying the above formulas, we are able to transform uniformly distributed random variables x_1 and x_2 to two random variables y_1 and y_2 with a normal distribution.

3.4 Linear Transformation

A linear transformation between two vector spaces U and V is a mapping $T : U \to V$ such that

1. $T(u_1 + v_2) = T(u_1) + T(u_2)$, for any vectors u_1 and u_2 in U,
2. $T(\alpha u) = \alpha T(u)$, for any scalar α and arbitrary vector u in U.

Suppose $U = R^2$ and $V = R^2$, $T : R^2 \rightarrow R^2$ is a linear transformation if and only if there exists a 2×2 matrix A such that $T(u) = Au$ for all u in R^2 [7]. Matrix A is called the standard matrix for T. Linear transformation in two dimensional vector space has been extensively used in our data generation system to dynamically produce two dimensional datasets of various characteristics. Once we have obtained the *basic dataset*, which will be detailed in section 4, we can control the shape, density and location of each cluster in the output dataset by applying to each vector/point in the basic dataset linear transformations such as shears, reflections, contractions, expansions and translations. The linear transformation of a normal distribution is still a normal distribution, but the linear transformation of a uniform distribution is not necessarily a uniform distribution.

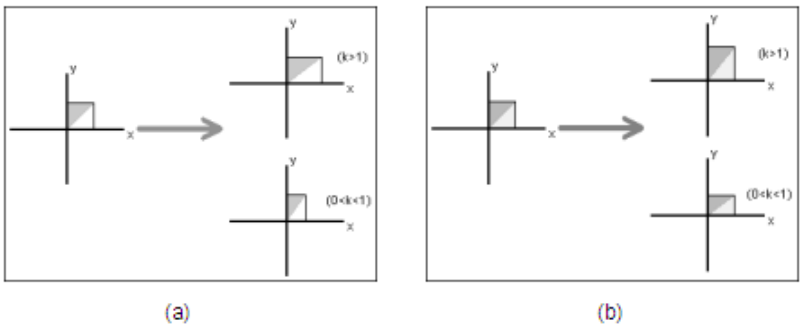


Fig. 3. Linear transformation: expansions and contractions

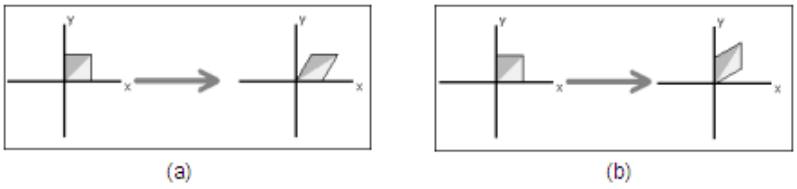


Fig. 4. Linear transformation: shears

Figure 3 and 4 are examples used in [7] to illustrate the action of a linear transformation $T : R^2 \rightarrow R^2$. The image of a unit square under T is employed to demonstrate the geometric meaning of different types of linear transformation.

Figure 3(a) indicates how expansion and contraction along x-axis work. Given a set of column vector $[p_x \ \ p_y]^T$, expansion and contraction along x-axis is given by the standard matrix

$$A = \begin{bmatrix} k & 0 \\ 0 & 1 \end{bmatrix}.$$

Thus, the vectors "stretch" along the x-axis to $[kp_x \ \ p_y]^T$ for $k > 1$ and "compress" along the x-axis for $0 < k < 1$.

Similarly, Figure 3(b) is an example showing the expansion and contraction of the unit square along y-axis. The standard matrix used here is

$$A = \begin{bmatrix} 1 & 0 \\ 0 & k \end{bmatrix},$$

which takes the vectors $[p_x \ \ p_y]^T$ to $[p_x \ \ kp_y]^T$. In this case, the standard matrix A stretches the vector along y-axis when $k > 1$ and compresses it along y-axis when $0 < k < 1$.

A shear in the x-direction is shown in Figure 4 (a). It is achieved using the standard matrix

$$A = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix},$$

to convert vectors $[p_x \ \ p_y]^T$ to $[(p_x + kp_y) \ \ p_y]^T$.

A shear in the y-direction is given in Figure 4 (b), in which the standard matrix

$$A = \begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix},$$

is used taking $[p_x \ \ p_y]^T$ to $[(p_x + kp_y) \ \ p_y]^T$.

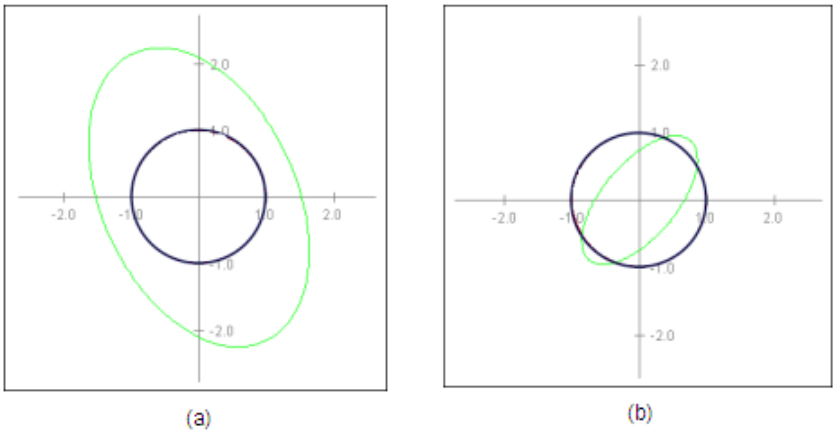


Fig. 5. Linear transformation examples

To generate datasets with various patterns and densities, we often use a more complicated standard matrix to transform a set of data. The operation can be considered as the composition of several linear transformation, such as a rotation, a magnification, and a translation. A typical example is shown in Figure 5 where a unit circle is transformed into an enlarged oval as in (a) and a contracted oval as in (b), where the standard matrices leading to these transformations are

$$A = \begin{bmatrix} -1.2 & 1.1 \\ 2 & 1 \end{bmatrix},$$

and

$$A = \begin{bmatrix} 0.3 & 0.8 \\ 0.9 & 0.3 \end{bmatrix}$$

respectively. The transformed ovals may be shifted to a different location by translations through vector addition.

4 A Comprehensive Approach to Synthetic Data Generation

In this section, we present a hybrid approach to synthetic data generation. The proposed approach is aimed at providing a basic modelling framework for generating data that can be used to evaluate and test clustering and outlier analysis algorithms.

It has been well recognized that the performance of different data analysis algorithms depends heavily on the testing datasets. Among the existing clustering algorithms, the partitioning methods can easily identify clusters with spherical shapes, but they are unable to find clusters of irregular shapes and tend to split an elongated cluster into different groups. Although the density-based methods can handle clusters of arbitrary shapes and various sizes, they are very sensitive to the density of each cluster, which will lead to failure in detecting clusters with data unevenly distributed. Since outliers are data that deviate from the main pattern of a dataset, they are always considered in the context of clusters. That is, an object is marked as an outlier if it is isolated from the clusters in the dataset. The causes for such isolation can be generalized in two categories: (1) outliers are located in a less dense region compared to the density of the clusters; and (2) outliers do not fit into the cluster patterns. Therefore, outlier detection, especially local outlier detection that defines outliers with respect to the neighborhood density and patterns is often conducted by differentiating them from data in clusters. A recent study [3] also shows that some outlier detection and clustering analysis algorithms are actually complementary to each other.

In our method of synthetic data generation, each output dataset is specified by a difficulty level, which is defined in terms of data distributions and cluster shapes. Since the difficulty level of a dataset indicates the complexity in identifying clusters, it provides us a measure of how a clustering algorithm works. Apart from the difficulty level, each dataset is also assigned a density level and

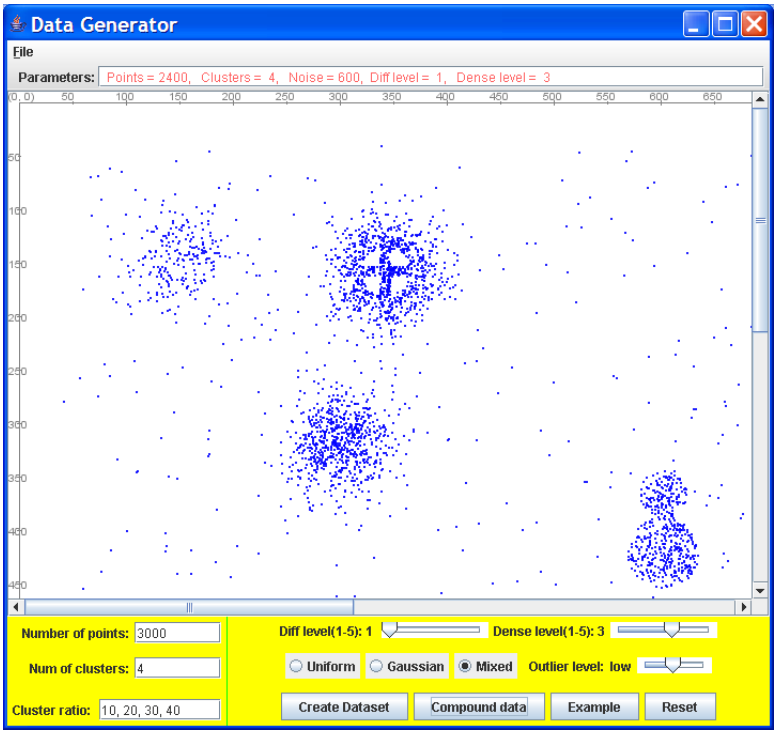


Fig. 6. A screen shot of the synthetic data generation system

a noise level. Like the difficulty level, noise level is used to define the spreading and density of outliers or noise. Other parameters from user input but controlled automatically by the system are the number of points, the number of clusters and the percentage of points for each cluster in a dataset. The created data points are in two dimensional space with x and y being floating point numbers. We built a graphical user interface to display the generated dataset for visual inspection. Figure 6 is a screen shot of the synthetic data generation system. As is seen in the figure, the shape and density of the output clusters as well as the distance between the means of different clusters in a dataset are determined by the standard distribution, the difficulty level and the density level.

To automate the data generation process, the system proceeds in two steps. The first step is to create the *basic dataset*, in which the data in each cluster have a standard distribution. For uniform distribution, the x and y values of all the points in the basic dataset are in $(0, 1)$. For normal distribution, the basic dataset contains clusters that have a standard normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$. The second step is to apply some mathematical techniques to generate the required dataset. Once we have the basic dataset, three major methods are used in creating clusters and outliers with different shapes and densities.

- Linear transformation, which involves matrix multiplication to translate, shear, contract or expend the the clusters in the basic dataset.
- Linear equation, which controls the line-shaped clusters and outliers.
- Circle equation, which controls the curve-shaped clusters.

The technical details of synthetic data generation will be presented in two aspect. One is the dynamic control and generation of clusters. The other aspect is about outliers, that is how the outliers are distributed. To make the concept concrete to the readers, a visual approach is taken in presenting the method.

4.1 Generation of Clusters in a Dataset

The generation of clusters of a dataset involves the determination of cluster densities, sizes, shapes and relative locations. By analyzing the user input, the synthetic data generation system automatically controls all these aspects. Two parameters *density level* and *cluster ratio* are the major factors to contribute to the density and size of each cluster in a dataset. Given a density level, an appropriate standard matrix is calculated to transform the basic clusters [1] into ones with either expanded or contracted sizes. The higher the density level, the smaller the cluster size and the more compacted the data in the clusters. By default, data are evenly distributed to each cluster in a dataset. For example, if dataset D has 1,000 data objects that forms 4 clusters, the system would automatically assign 250 data to each cluster. The parameter *cluster ratio* provides the user with an option to set the number of data objects for each cluster. It

[1] attribute values in such clusters are usually in $(0, 1)$

consists of a sequence of integers indicating the percentage of data in each cluster over the total number of data in a dataset. By parsing the cluster ratios, the system adjusts the number of data in each cluster to satisfy the user's specific requirements. This, in turn, will change the density of each cluster since each cluster size remains unchanged.

Cluster shapes and relative locations are mostly determined by the parameter *difficulty level*. In the following, we will discuss the generation of datasets classified into five difficulty levels based on the distribution of the data in clusters. Given a difficulty level, the specific locations and shapes of the clusters in a dataset is controlled by the system in a random manner, i.e., the cluster can have any of the shapes belonging to this difficultly level and lie in any region in the dataset. The distances between clusters are dynamically measured to ensure clusters are not overlapping, which is especially important for simple datasets with low difficulty levels. Alternatively, a dataset may consist of randomly produced clusters from different difficutly levels when one prefers to have a sophisticated set of data. Therefore even with identical parameter sets, there are hardly any datasets that are exactly the same due to the randomness in deciding cluster locations and shapes. Apart from being visualized, the generated data can be saved to a file in case that the same data are required for later inspection or testing different data analysis algorithms.

Datasets with Difficulty Level One

The datasets at this level are the simplest in terms of the definition of clusters. There are two major features of the clusters in such a dataset.

- All clusters have only spherical or square shapes.
- Clusters are well separated.

Following the generation of the basic datasets, the transformation of contraction and/or expansion are applied to generate the datasets that satisfy the user-specified density level. Figure 7 shows the typical clusters in a dataset hav-

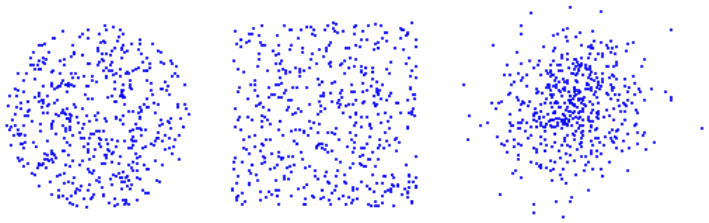


Fig. 7. Difficulty level 1: each cluster contains 500 2D points

ing a difficulty level of one. It can be seen that such design of data distribution

ensures that data are clearly divided into well-formed groups which makes it relatively easy for clustering algorithms to find the clusters. When evaluating clustering methods with these type of datasets, we are mostly concerned about how fast a certain method can identify the clusters in a large dataset.

Datasets with Difficulty Level Two

The datasets have long and thin clusters with straight or curved shapes. Like clusters in level one, clusters in a particular dataset are well separated. Figure 8

Fig. 8. Difficulty level 2: each cluster contains 300 2D points

gives some of the example clusters in the datasets having a difficulty level of two. Based on the input parameters, linear equations and transformations of contraction, expansion, rotation and translation are performed on the basic dataset to create level-two datasets. Although the clusters are at an easy level and are as intuitive as the first level ones, their enlongated shape can make some clustering methods fail in identifying them. For example, the algorithms k-means [11] and k-medoids [10] are most likely to split such a cluster into two or more groups as they favor spherical shaped clusters.

Datasets with Difficulty Level Three

The clusters in the dataset with difficulty level three have simple arbitrary shapes such as rings, crosses and stairs. The distance between the clusters are clearly separated. Some typical clusters are given in Figure 9 in which the three clusters on top have uniform distributions and the two at the bottom have normal distributions. In order to generate these datasets, linear transformations such as contraction, expansion, rotation and translation as well as linear equations and circle equations have been performed on the basic dataset of either uniform or normal distribution.

Compared to the first two level datasets that contain only basic convex clusters, the level-three datasets have clusters that do not necessarily have an object

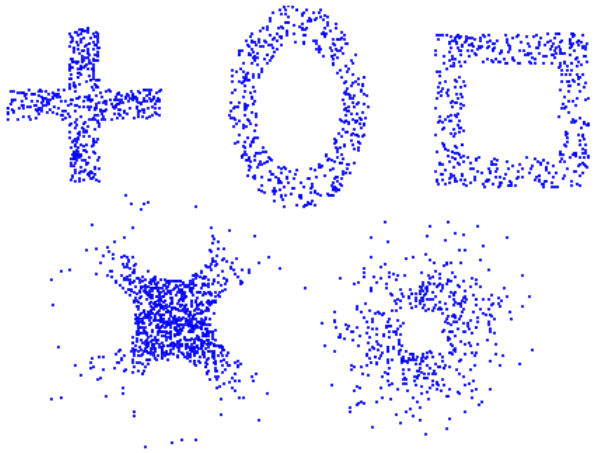


Fig. 9. Difficulty level 3: each cluster contains 500 2D points

defined as the mean. For example, there is not any object that can be considered as the explicit mean for a ring shaped cluster. Consequently, the irregular shape of clusters will increase the difficulty in finding meaningful clusters for any clustering algorithm that considers the mean.

Datasets with Difficulty Level Four

The clusters in the dataset with difficulty level four have arbitrary shapes with some obvious or vague space inside a cluster. The distance between the clusters are still distinguishable. To enrich the diversity of cluster shape, clusters with uniform distribution are specifically designed to be any of the twenty-six letters of the alphabet which are evenly positioned in a particular dataset. Each letter is treated as an individual clusters. The system provide two options for generating the required number of alphabet clusters. One is to randomly produce any of the letters. The other option allows the user to input letters of his own interest. The operations used to control the distribution and shape of the letters involve all the techniques mentioned including equations and transformations. Example clusters are shown in Figure 10 in which the letters have uniform distributions and the other two clusters have normal distributions. Since the letters encompass a wide range of cluster shape, it is hard for most clustering methods to find all the different letter-shaped clusters. Although density-based algorithms such as DBSCAN can work well with datasets containing diverse cluster shape, they will fail in identifying some of the clusters if the densities between clusters are quite different.

Fig. 10. Difficulty level 4: each cluster contains 500 2D points

Datasets with Difficulty Level Five

The datasets contain clusters within clusters or single clusters with irregular shapes. In the case of one cluster within the other cluster, the two clusters can either be clearly separated or they are connected with bridges of points, which can cause much trouble to many clustering algorithms in correctly identify the clusters. Nested clusters also raise a question as to how to define a cluster: should we consider a nested cluster as one cluster or several clusters? Figure 11 displays some of the clusters in datasets having a difficulty level of five. Besides the generation mechanism for creating clusters of the other levels, special attention is paid to positioning the nested clusters at this level.

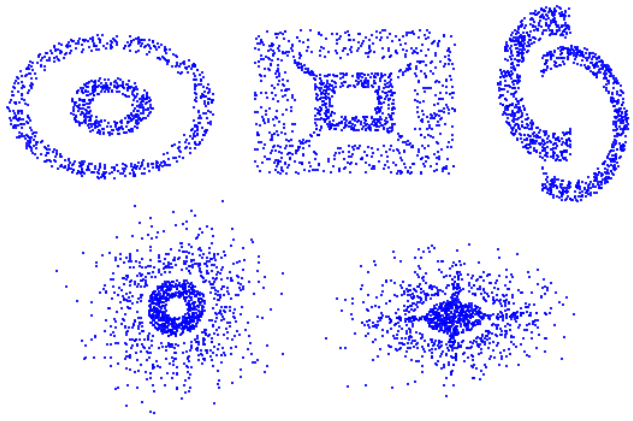


Fig. 11. Difficulty level 5: each paired cluster contains 1,000 2D data points, of which 500 are assigned to each single cluster

4.2 Generation of Outliers/Noise in a Dataset

As discussed in the previous chapters, outliers and noise are unavoidable in real life datasets collected from numerous application domains. The mechanism of adding outliers or noise is another important contribution of our synthetic data generation system. While there is no strict distinction between outliers and noise in most data analysis tasks, we will use outliers as a generic term in the following discussion.

It is well accepted that outliers in a dataset are not consistent with the rest of the data. This leads to the exploration of outlier detection based on the distance to a point's neighboring points. Many existing outlier detection methods use the neighborhood density of a point as a criterion to differentiating abnormalities from normalities. Points located in a less dense region are usually considered as outliers. While intuitive, such definition raises new issues: how do we specify the cutoff density value to guarantee real outliers and meaningful clusters? Should the points located in the outer layers of a normal distribution as shown in Figures 7 through 11 be marked as outliers? Or should all the data in a less dense cluster be treated as outliers?

Because the definition of outliers is subjective, the notions of outliers and inliers in a dataset are ambiguous in many situations. Data object being identified as outliers by one data analysis method could be legitimate inliers with the other method. Therefore, To produce outliers with respect to local and global clusters, our effort is focused on how to generate those data points that can be objectively identified as outliers by the existing outlier detection algorithms.

The method of generating outliers is similar to that of generating clusters. Standard distribution and linear transformation have been widely used. The distribution and density of outliers are determined by the system through the parameter: outlier level. The value of outlier level can be none, low and high and are specified by the user. Depending on the selected level, the number of outliers is a controlled percentage of the total number of points in a dataset. For example, the outliers account for 10% of the data when 90% of the data are in clusters. This ensures that the total number of points from the user input is preserved while outliers are being added. Next we will discuss the generation of the three level outliers. The examples used are all sophisticated datasets that contain clusters of different difficulty levels.

Outliers Level None

The name of "level none" is self-explaining. No outliers are intentionally added to a dataset. However, this does not necessarily mean that a set of generated data does not contain outliers. A dataset often consists of clusters with different distributions and densities. Depending on the definition of a specific outlier detection algorithms, data points in clusters of different difficulty levels as described before can be outliers. For example, a cluster itself can be considered as a collection of outliers if the size of the cluster is much smaller than those of other clusters or the data in the cluster are very sparsely distributed compared

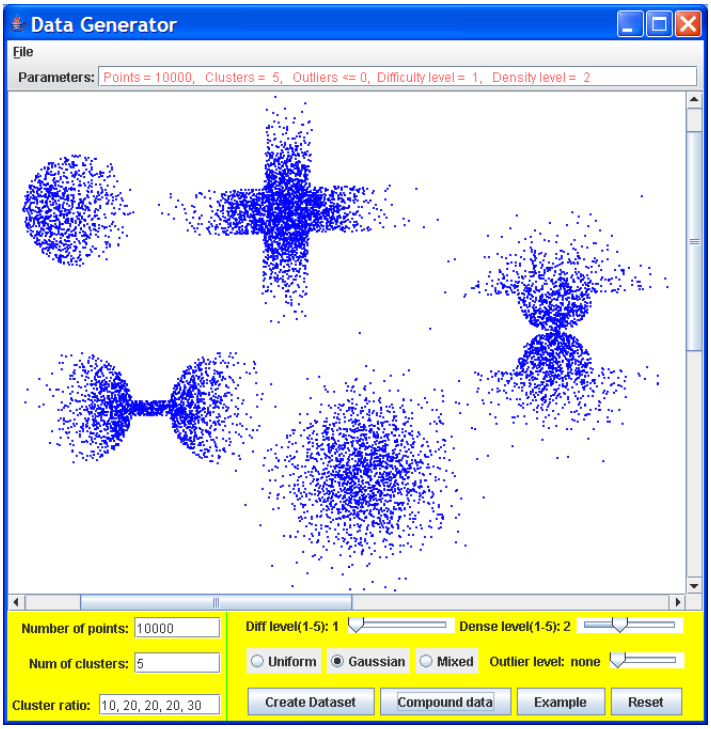


Fig. 12. Outliers level none: outliers are those exterior points of a cluster

to the majority of the data. Most outlier detection algorithms would mark the exterior points in a normally distributed cluster as outliers. Such examples are demonstrated in Figure 12.

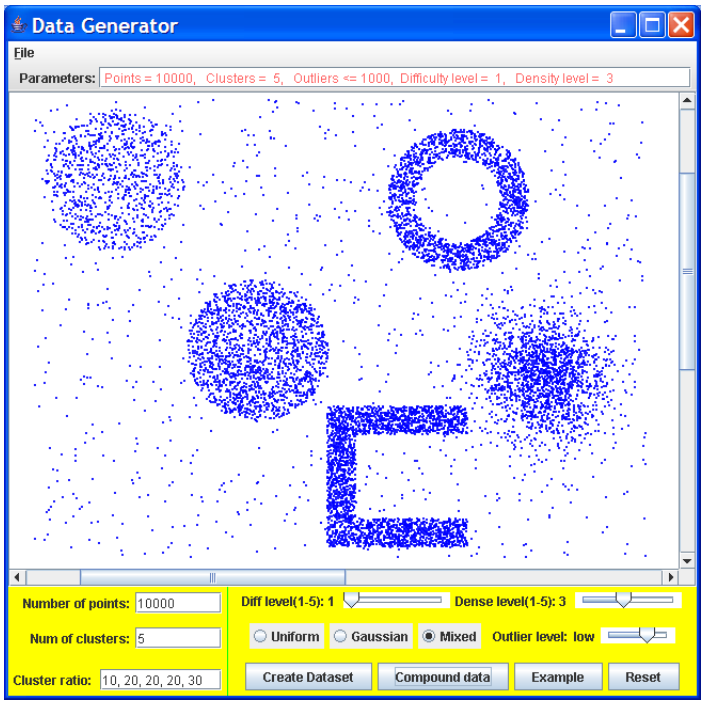


Fig. 13. Outliers level low: outliers are randomly distributed

Outliers Level Low

This is the basic type of outliers. For a given dataset, the system randomly distributes a small percentage of the data in the region where the clusters are located. Figure 13 shows a dataset containing 4,000 points including outliers.

Outliers Level High

In addition to generating randomly distributed outliers, the data generator produces outliers of controlled shape and distribution. Since there is no universal agreement on what constitutes outliers, our intention is to provide a prototype of outlier distribution in a dataset. Figure 14 gives an example dataset containing 5,000 2D points in which outliers count up to 15% of the total data. Two major types of data points can be classified as outliers in this dataset:

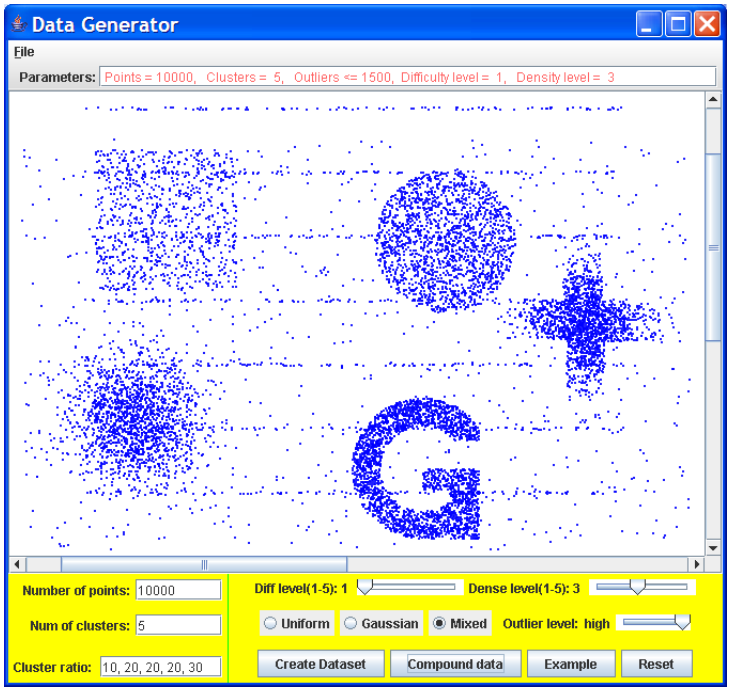


Fig. 14. Outliers level high: outliers are either randomly distributed or have simple patterns

1. points that are located in a sparse neighborhood;
2. exterior points of the cluster that has a normal distribution; and
3. points that form certain patterns, such as the lines, each of which has much less data than those major clusters. Some may not consider these points as outliers because they form a major pattern. Depend on the density of the lines, these points can be classified into either points in clusters or outliers. We will demonstrate this in section 5 with experimental results.

Many clustering and outlier analysis algorithms can easily identify the first two type of outliers that have sparse neighborhoods. But the third type of outliers can cause problems in the process of data clustering. For example, the density-based clustering algorithm DBSCAN has been well recognized as an effective method in finding clusters of arbitrary shapes as well as identifying and eliminating outliers. However, it may merge two or more clusters together as the lines or the so-called bridges of points join these clusters into a group.

5 Experiments and Evaluation

One of the most effective ways to evaluate the generated dataset is to visualize the data for human inspection. The GUI of the data generating system has been designed to serve this purpose. In addition to visual inspection, we test the performance of our system in two aspects:

- the efficiency of producing large datasets that satisfy user's requirement;
- the effectiveness of a benchmark instance generator for clustering analysis and outlier detection.

In this section, we report experiments and evaluation results of our synthetic data generation system.

5.1 Generating Very Large Datasets

We first test how the size of the generated datasets affects the execution time. For any dataset with size up to 1,000,000 points, the execution time for generating the data (excluding writing the data to a file) is less than three seconds. This is demonstrated in Figure 15, which is a plot of the execution time against the size of the generated datasets. It is observed that to generate a dataset containing less than 4,000,000 data points, the execution time is linear to the size of the dataset regardless of difficuty levels, density levels and outlier levels.

Since the difficulty level is the major factor that determines the distribution and shape of each cluster in a dataset, we also ran the program to show how the execution time is affected by the difficulty level. For each difficulty level (from 1 to 5), we input the same parameters which include data size, number of clusters, density and outlier levels so that the difference of data generating time is exclusively based on the change of difficulty levels. Despite the same parameters, each dataset produced may contain clusters of different distributions, shapes and

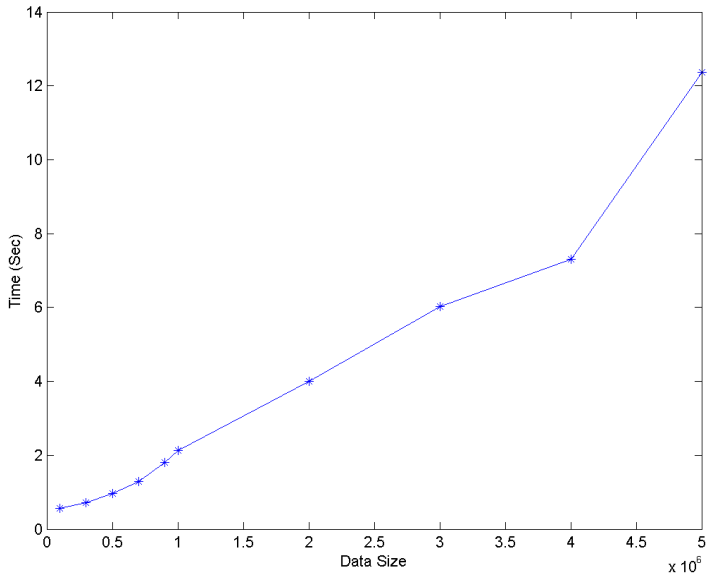


Fig. 15. Each generated dataset has the following properties: number of clusters is 5; data distribution in a cluster is either uniform or normal; difficulty level ranges from 1 to 5, density level is 3, and noise level is low

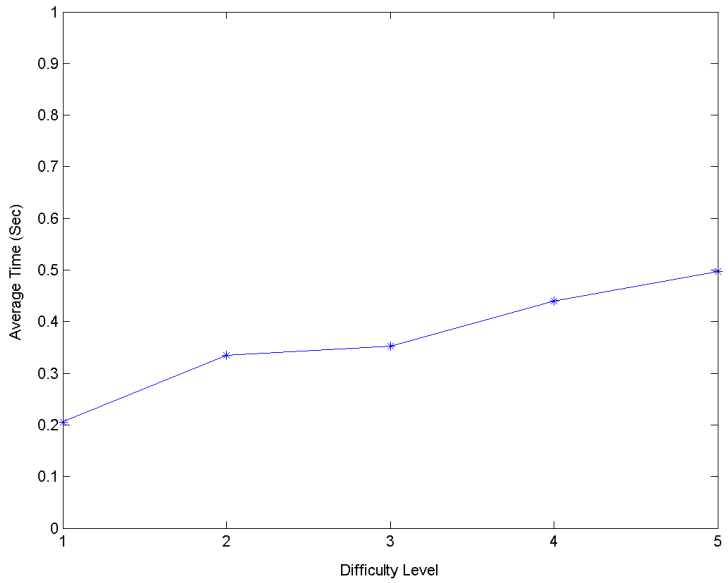


Fig. 16. With each difficulty level, the system generates a dataset of 100,000 that contains both uniformly and normally distributed clusters.

densities. In order to precisely show the execution time of generating a dataset, we ran the program at least five times for each difficulty level and then computed the average excution time. The plot in Figure 16 demonstrates that with the change of difficulty levels, there is little change of average execution time to generate a certain type of dataset.

5.2 Testing with Clustering and Outlier Analysis Algorithms

We generated six sets of two dimensional spatial data. Each dataset contains outliers as well as clusters that consist of either uniformly or normally distributed data. The details of the datasets are given in Table 1. Clusters in each of the first five datasets exhibit the typical cases of data distributions and shapes of a specific difficulty level. The sixth dataset, however, contains a mixture of clusters that are randomly generated from different difficulty levels. The sizes of the datasets are moderate for easy inspection and illustration.

Table 1. Detailed description of the parameters for the datasets

Dataset	**Data size**	**Number of clusters**	**Difficulty level**	**Noise level**
dataset1	2,000	4	1	low
dataset2	2,000	4	2	low
dataset3	2,000	4	3	low
dataset4	2,000	5	4	high
dataset5	2,000	5	5	high
dataset6	10,000	7	mixed	high

Table 2. Description of the clustering algorithms

Algorithm	**classification**	**Parameters**
k-means	partition-based	k
DBSCAN	density-based	radius ϵ, $MinPts$
CURE	hierarchical	k, shrinking factor α, representative points t
CHAMELEON	hierarchical	$k - NN$, $MinSize$, k
WaveCluster	grid-based	resolution r, signal threshold τ
AutoClass	model-based	N/A

Using these datasets as benchmark instances, we conducted experimental evaluation upon six existing clustering algorithms: k-means [10], DBSCAN [5], CURE [6], CHAMELEON [9], WaveCluster [15] and AutoClass [1, 2]. The CURE code is kindly supplied by the Department of Computer Science and Engineering, University of Minnesota. The AutoClass is the public C version from [20]. The

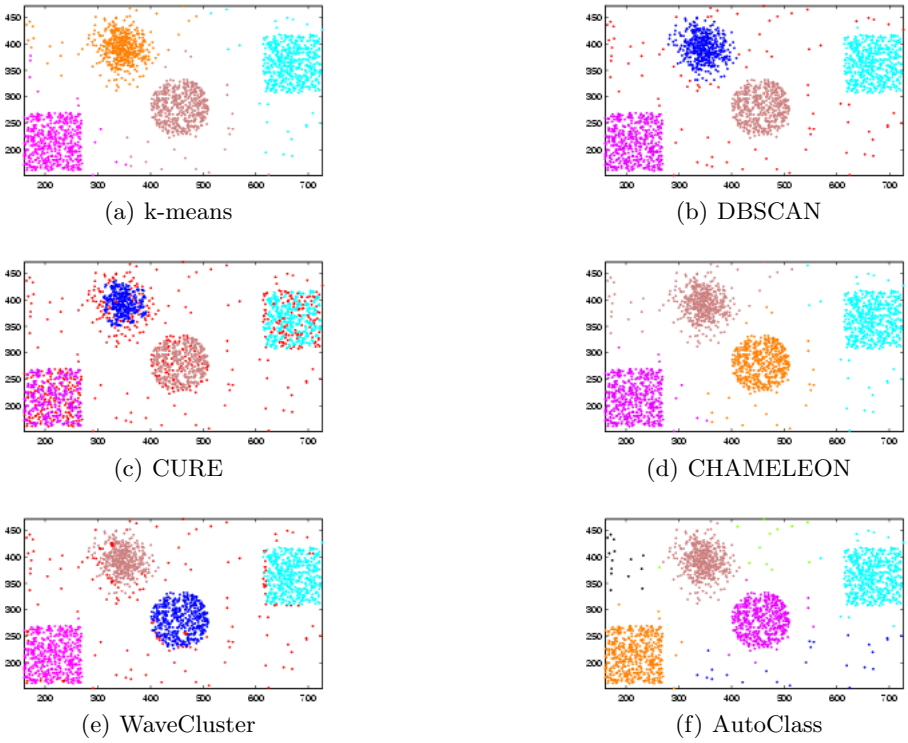
other four programs were locally implemented. Some basic characteristics of these clustering methods are generalized in Table 2.

Our experiments proceed from *easy* datasets to *hard* datasets. The complexity of a dataset is defined by the difficulty levels as described in the previous section. Our intension is not to explore the different clustering mechanisms, instead, we aim to show experimentally how each of these six clustering algorithms performs with different datasets consisting of a diversity of clusters and difficulty levels. We show the clustering results on each dataset graphically to give readers a concrete idea of the clustering ability of different clustering methods. We assume that we have the specific domain knowledge of each dataset. When performing the experiment, such domain knowledge plays an important role in the selection of certain parameters, such as k, the number of clusters involved in some of the algorithms. To avoid the bias of inappropriate use of other parameters for different algorithms, we also conduct many test-and-trials to select the set of parameters that lead to the best clustering results of the algorithm being tested.

In Figure 17 to 22, different colors have been employed to indicate discovered clusters in a dataset after the clustering process. Since some of these clustering methods, such as DBSCAN, CURE and WaveCluster, are able to identify outliers, red color is reserved to mark outliers in all the clustering results of the following figures. Figures 17 to 22 can be viewed in two ways:

- Given a certain dataset, inspect the clustering abilities of different clustering algorithms, and
- For a certain clustering method, check its clustering results over different sets of data.

The definition of clusters and outliers is often subjective. Meaningful clusters and real outliers should be considered in the context of application domains. Even with the synthetic datasets used in our experiments, it is sometimes not easy to mark clusters from clusters or to distinguish clusters from outliers. For example, in figure 21, CURE and AutoClass treat the diagonal line pattern as a single cluster while other methods consider it either as part of another cluster or as outliers. Another example is the clustering results shown in 22 obtained from dataset 6, where the small oval and big rectangle (cluster in cluster) are grouped into one cluster by all the six clustering methods although they might well be considered as two clusters. Pros and cons of various clustering algorithms have been widely discussed in the literature, we evaluate these algorithms based on the quality of the clustering results on the given datasets.

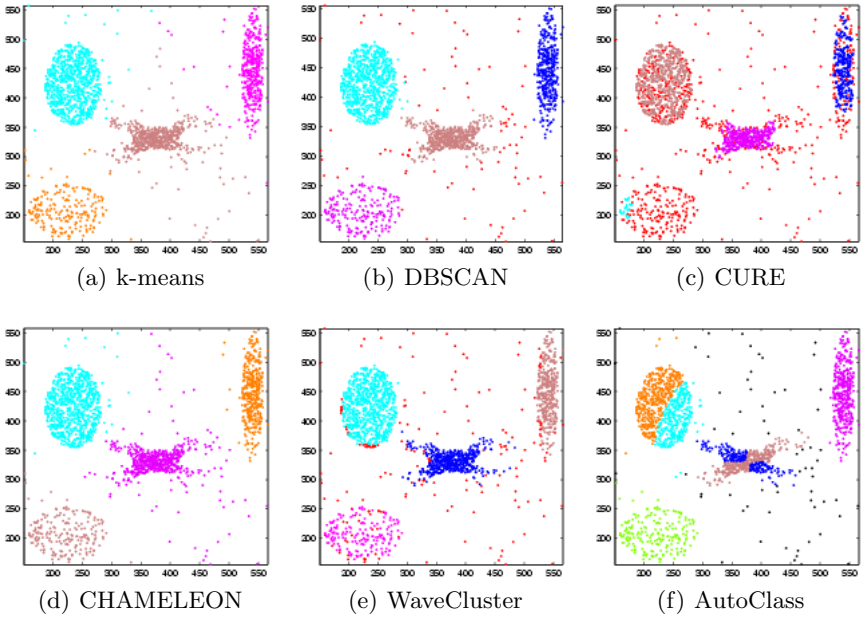


(a) k-means (b) DBSCAN

(c) CURE (d) CHAMELEON

(e) WaveCluster (f) AutoClass

Fig. 17. Clustering results on dataset 1. (a): k-mans with $k = 4$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 4$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k - NN = 15$, MinSize=2.5%, and $k = 4$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass



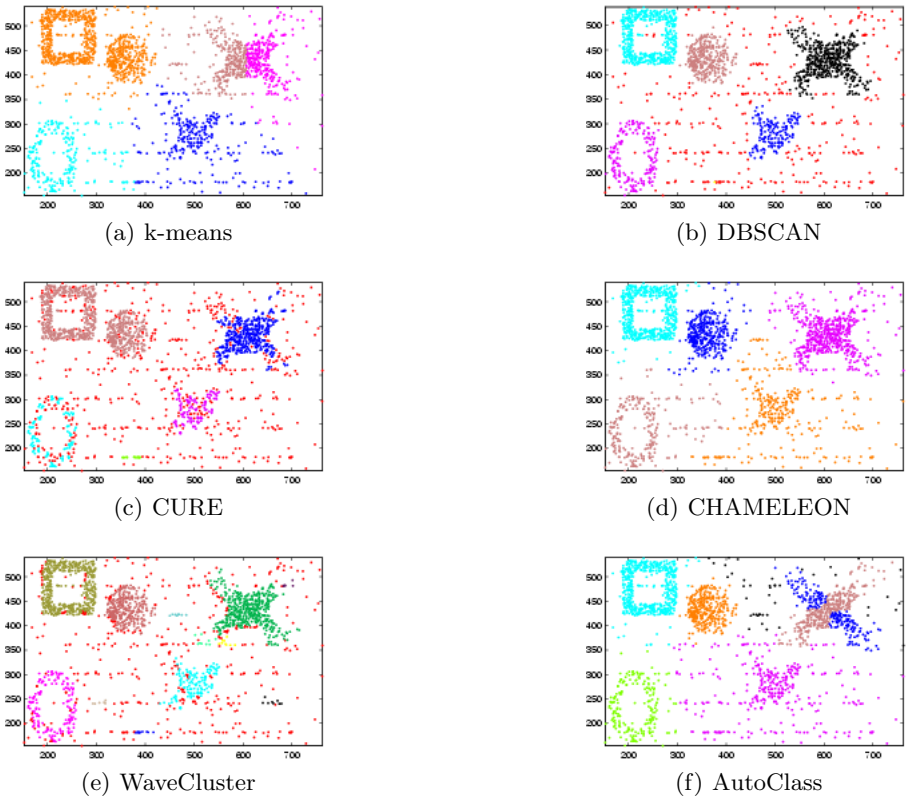
(a) k-means (b) DBSCAN (c) CURE

(d) CHAMELEON (e) WaveCluster (f) AutoClass

Fig. 18. Clustering results on dataset 2. (a): k-mans with $k = 4$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 4$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k - NN = 15$, MinSize=2.5%, and $k = 4$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass

Some interesting observation from the experiments can be generalized as follows.

1. K-means is well known for being able to quickly find spherical shaped clusters. Through the experiments on datasets of different levels, it is found that k-means can successfully identify irregular shaped clusters if the distances between clusters are big enough and the initial set of centroid have been well selected. Three major factors that mostly affect the clutering results of k-means are: (1) domain knowledge for the selection of parameter k; (2) initial location of the set of centroid; and (3) distribution of outliers.
2. Given the appropriate values for the two parameters: neighborhood radius ϵ and MinPts, DBSCAN achieves the best clustering results among the six algorithms. It can not only find arbitrary shaped clusters but can also detect most outliers. One intrinsic shortcoming of DBSCAN is that it may merge two or more clusters if there exist "bridges" of outliers joining clusters such as Figure 21 (b).
3. CURE is designed to not only find arbitrary-shaped clusters, but also identify outliers in a dataset. Our experiments indicate that CURE can successfully find meaningful clusters that have identical densities, but it also marks many points that located in the uniformly distributed clusters as outliers as

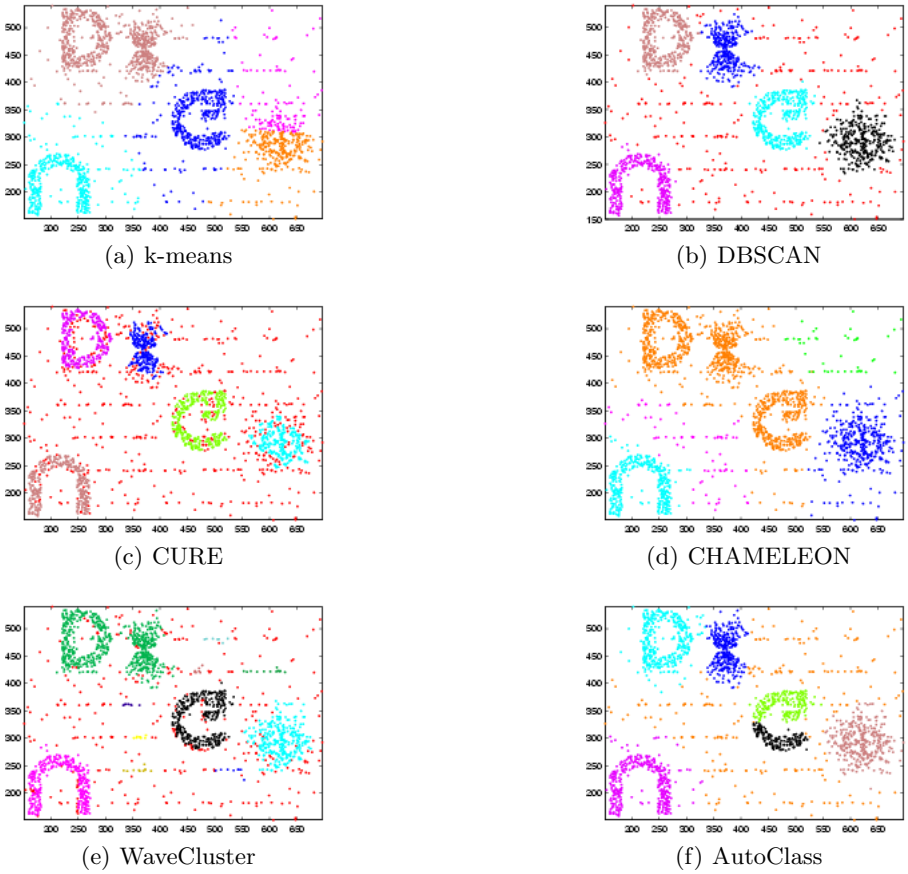


(a) k-means (b) DBSCAN

(c) CURE (d) CHAMELEON

(e) WaveCluster (f) AutoClass

Fig. 19. Clustering results on dataset 3. (a): k-mans with $k = 5$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 5$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k - NN = 15$, MinSize=2.5%, and $k = 5$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass

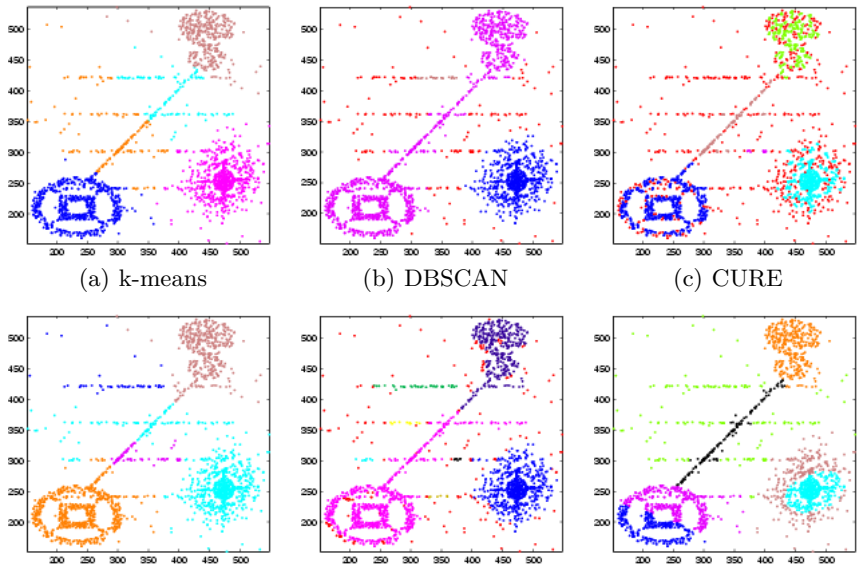


(a) k-means (b) DBSCAN

(c) CURE (d) CHAMELEON

(e) WaveCluster (f) AutoClass

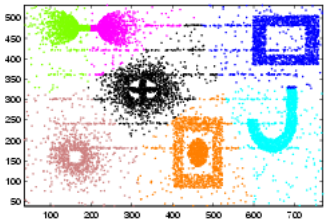
Fig. 20. Clustering results on dataset 4. (a): k-mans with $k = 5$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 5$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k - NN = 15$, MinSize=2.5%, and $k = 5$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass



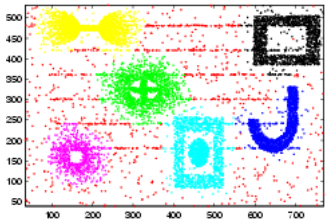
(a) k-means (b) DBSCAN (c) CURE

(d) CHAMELEON (e) WaveCluster (f) AutoClass

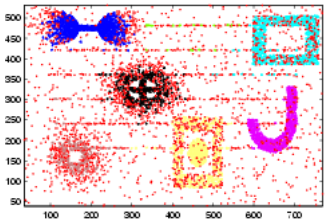
Fig. 21. Clustering results on dataset 5. (a): k-mans with $k = 5$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 5$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k - NN = 15$, MinSize=2.5%, and $k = 5$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass



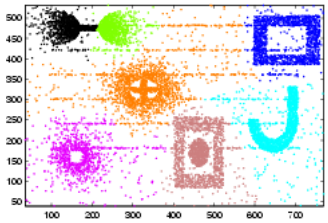
(a) k-means



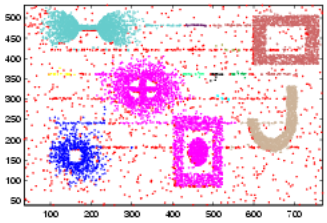
(b) DBSCAN



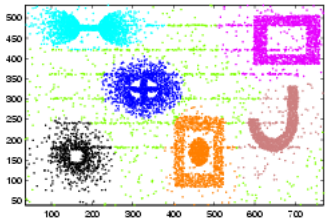
(c) CURE



(d) CHAMELEON



(e) WaveCluster



(f) AutoClass

Fig. 22. Clustering results on dataset 6. (a): k-mans with $k = 5$; (b): DBSCAN with $\epsilon = 20$ and $MinPts = 30$; (c): CURE with $k = 7$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k - NN = 15$, MinSize=2.5%, and $k = 7$; (e): WaveCluster with $r = 4$ and $\tau = 0.2$; (f): AutoClass

demonstrated with all the six datasets. A big problem with CURE is that it might fail to find some real clusters when the densities of these clusters are relatively less than those of the other clusters in a dataset as shown in Figure 18 (c).

4. Like k-means, CHAMELEON can not handle outliers. Although it is extremely slow in performing the clustering, it is more effective than k-means as it can find clusters of arbitrary shapes regardless of the distances between clusters.
5. In most cases, WaveCluster is effective in finding clusters and outliers in a dataset. Although the number of resulted clusters is often more than the actual number of clusters in a dataset as shown in Figure 19 (e), 20 (e), 21 (e) and 22 (e), major clusters usually stand out since they contains far more data objects than those small clusters. A further step to eliminate small clusters and mark the data objects in these clusters as outliers would surely improve the effectiveness of WaveCluster.
6. The most interesting clustering algorithm used in our experiments is AutoClass. It is an unsupervised Bayesian classification system that seeks a maximum posterior probability classification [20]. Such method has been widely used in statistics and machine learning. The uniqueness of AutoClass is that it can find data clusters that might not be identified as clusters by visual inspection. For example, the blue clusters in Figure 19v(f) and Figure 20v(f). This is due to the fact that AutoClass is able to find clusters that is maximally probable with respect to the underlying data model. Though not designed to identify outliers, AutoClass can generally classify outliers into one group even though they are usually separated by clusters.

6 Conclusion

In this chapter, we present a comprehensive approach to synthetic data generation for data analysis and demonstrate that the approach is very effective in generating testing datasets for clustering and outlier analysis algorithms. According to the user requirements, the approach systematically creates testing datasets based on different data distribution and transformation. Given the number of points and number of clusters, each dataset is controlled by data distribution, difficulty level, density level and outlier level. The difficulty level determines the overall characteristic (shape, position) of the clusters in a dataset, the density level mostly determines the size and density of each cluster. The generated datasets contain clusters of two standard distributions: uniform distribution and/or normal distribution. While the synthetic data generation system is effective in generating two-dimensional testing datasets to satisfy user's requirement, it is proven to be efficient in generate very large dataset with arbitrary shaped clusters. The current object-oriented system is carefully designed so that it can be easily extended to handle data in high-dimensional space in the future.

7 Future Work

Synthetic data generation is an interesting topic in data mining. In many research areas, a set of standard dataset is essential in evaluating the quality of a proposed technique. Methods of generating datasets for different purposes can be quite different. Our work concentrates on the generation of test instances for clustering and outlier analysis algorithms. There are still much room for improving the current data generating system.

- The data generator now can only handle two-dimensional data. Based on the heuristic devised, the system can be extended to generate three or higher dimensional data.
- The size of a cluster is controlled by the density level, which ensures that the number of points in a cluster be fixed, but also poses a problem, i.e., clusters with a specific density have basically the same size. Finding a better way to address this problem can produce various sized clusters in a dataset.
- The current interface is very basic, further work is needed to improve the look and feel of the interface.
- In another more or less theoretical direction, it would be interesting to discuss the meaning of the difficulty of the datasets.

References

1. P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: A bayesian classification system. In *PProceedings of the Fifth International Conference on Machine Learning*, pages 54–56. Morgan Kaufmann Publishers, June 1988.
2. P. Cheeseman and J. Stutz. Bayesian classification (autoclass): Theory and results. In U. Fayyad, G. Paitesky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press, 1995.
3. Z. Chen, A. Fu, and J. Tang. On complementarity of cluster and outlier detection schemes. In *Proceedings of 5th International Conference on Data Warehousing and Knowledge Discovery, (DaWaK)*, pages 3–5, September 2003.
4. J. Edvardsson. A survey on automatic test data generation. In *Proceedings of the Second Conference on Computer Science and Engineering in Linkping (CCSSE'99)*, October 1999.
5. M. Ester, H-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.
6. S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *In Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 73–84, 1998.
7. HMC. Geometry of linear transformations of the plane. Internet page. `http://www.math.hmc.edu/calculus/tutorials/`.
8. IBM. Intelligent information systems. Internet page. `http://www.almaden.ibm.com/software/quest/resources/`.
9. G. Karyapis, E.-H. Han, and V. Kumar. CHAMELEON: A hierarchical clustering algorithms using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.
10. L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons, 1990.
11. J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. University of California Press, 1967.
12. Mathworld. Box-muller transformation. Internet page. `http://mathworld.wolfram.com/`.
13. S. Ross. *A first course in probability.* Prentice Hall, 1997.
14. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2nd edition, 2003.
15. G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of 24th International Conference on Very Large Data Bases*, August 1998.
16. StatCrunch. Statistical software for data analysis on the web. `http://www.statcrunch.com/`.
17. Statlets. Statpoint internet statistical computing center. Internet page. `http://www.statlets.com/`.
18. Y. Theodoridis and M. Nascimento. Generating spatiotemporal datasets on the www. *ACM SIGMOD Record*, 29(3), September 2000.
19. VBRC. Viral bioinformatics resource center. Internet page. `http://athena.bioc.uvic.ca/techDoc/`.

20. T. Will. NASA ames research center: The autoclass project. Internet page. http://ic.arc.nasa.gov/ic/projects/bayes-group/.
21. Y. Yu, D. Ganesan, L. Girod, D. Estrin, and R. Govindan. Synthetic data geneartion to support irregular sampling in sensor networks. In *Geo Sensor networks*, October 2003.