SynDECA: A Tool to Generate Synthetic Datasets for Evaluation of Clustering Algorithms

Jhansi Rani Vennam Soujanya Vadapalli

Centre for Data Engineering
International Institute of Information Technology
Gachibowli, Hyderabad, INDIA
(jhansi,soujanya)@iiit.ac.in

ABSTRACT

A large number of clustering algorithms have been proposed of late, which can identify clusters of arbitrary shapes, varying densities and sizes. There are no benchmarking datasets with high dimensionality and noise, which can evaluate clustering algorithms on various aspects like scalability, accuracy and robustness to noise. Real-life datasets are few in number and do not have the "original" clustering results by default. This emphasizes the need to have a toolkit that can generate datasets, which mimic real-life data and provides the actual clustering results. In this paper, we propose few algorithms and methodologies that generate high-dimensional datasets along with the original clustering results. We developed a toolkit called SynDECA [15] that generates synthetic datasets based on the algorithms proposed.

1. INTRODUCTION

Clustering is the process of grouping a set of objects into classes of *similar* objects. A *cluster* is a collection of data objects that are similar to objects within the same cluster and dissimilar to those in other clusters [6]. Similarity [1] between two objects is calculated using a distance measure. The family of L_k-norm distances, Mahalanobis distance functions are few to mention. Grouping of the objects can be done using any of *partitioning* [7, 8], *hierarchical* [8, 14, 16], *density-based* [10, 9], *grid-based* [18] and *model-based* [1] techniques. The goal of clustering is to identify underlying patterns based on the similarities between the objects.

Of late, with the explosion of data, the size of datasets to be clustered has increased tremendously. Thus enforcing, the following new requirements on the clustering algorithms:

1. *Scalability of the algorithm:* The algorithm should be able to handle very large datasets. The order of the input records in a dataset should not affect the output of the clustering algorithm.
2. *High-dimensional data:* The algorithms should be able to handle large dimensional data.
3. *Heterogeneous attributes:* Ability to handle various types of attributes (numerical, categorical) is important, due to increasing nature of heterogeneous data, of late.
4. *Complex shapes of clusters:* The accuracy of the clustering result should be acceptable despite the presence of complex shaped clusters.
5. *Noise:* The algorithm should be able to handle noise too. Presence of noise should not deter the accuracy or efficiency of the clustering algorithm.

The most recent work in clustering algorithms [4, 3, 17, 5] address almost all the issues mentioned above. However, any clustering algorithm needs to be evaluated for its ability to handle these issues. For the sake of evaluation, we need datasets which are large, noisy and high-dimensional with the presence of complex clusters. In addition to this, the "actual" clustering result should also be available to benchmark the results given by various clustering algorithms. Such datasets along with the clustering results are very less in number. Though there are few readily available real-life datasets, the actual clustering results are not known. This necessitates a tool to be devised, capable of generating high-dimensional, noisy datasets along with the original clustering results. Our toolkit SynDECA (Synthetic Datasets to Evaluate Clustering Algorithms) generates large noisy high-dimensional datasets. Our techniques are robust and in our experimental section we show that SynDECA could generate a 100-dimensional dataset with 1,00,000 data points, with a small percentage of noise. SynDECA also provides the information about each point (whether it belongs to cluster or noise) and a brief statistical description of the clusters present in the dataset.

The outline of this paper is as follows. In section 2, we present the related work. Major steps approached in dealing

[1] The terms similarity and distance are used interchangeably. The less the distance between the two objects the more similar they are.

ADVANCES IN DATA MANAGEMENT 2005
Jayant Haritsa, T.M. Vijayaraman (Editors)

the problem are explained in section 3. In section 4, algorithms used to place clusters in the given space and generation of cluster points and noise are given. Section 5 gives the validity of the generated datasets. Few datasets that are generated are illustrated in section 6. Section 7 presents conclusions and future work.

2. RELATED WORK

The idea of synthetic dataset generation is not new and there have been attempts to generate synthetic data for use in various fields. For instance, an automatic test data generator [2] is one of the most important components in a testing environment. This kind of data generators contain a program analyzer, a path selector and a test data generator. Given a program P, which is represented as a control flow-graph and a path (unspecified) u, the aim is to generate input x, so that x traverses u. This can be done by finding the path predicate and then solving the path predicate in terms of input variables [2]. Basically three approaches are followed while constructing a test data generator *random, goal-oriented,* and *path-oriented.* Each of these methods can be implemented statically or dynamically.

In the area of spatio-temporal data management, datasets are required in order to evaluate spatio-temporal databases, spatio-temporal data modeling, query languages, spatio- temporal data mining and spatio temporal indexing. A recent work on data generation for spatio-temporal databases can be found in [12]. Given few parameters such as *duration*, *shift* and *resizing* of an object, tools like *GSTD, G-TERD, and Oporto* will generate spatio-temporal data. Out of which Oporto mimics a very specific scenario: fishing at sea. But these techniques being spatio-temporal, data is upto 4-dimensions.

Synthetic dataset generation is not only limited to program testing and spatio-temporal databases. There have been some early attempts to generate synthetic data to test the clustering algorithms. Few schemes for generating artificial data have appeared in the literature [11]. One of the processes used is multivariate normal mixtures with fairly complex covariance matrices. This leads to the generation of the non-overlapping clusters. The algorithm proposed by Glenn [11] generates data in either 4, 6, or 8 dimensional space containing up to 5 clusters. Three different methods are followed in assigning the points to each cluster.

The overall approach followed by the Glenn's algorithm is as follows. Initially the extent of each cluster in the first dimension is fixed in such a way that there will not be any overlap between any clusters in this dimension. Then the extent of each cluster in the remaining dimensions is calculated. Points are generated within the bounding box[2] of each cluster. Outliers associated for each cluster are generated, which are not within the bounding box of the cluster. Finally error measures, such as error perturbation to each dimension of each point in the dataset, are added. Since the clusters are non-overlapping in the first dimension, they remain non-overlapped in any number of dimensions. In such case any clustering algorithm will be able to identify clusters properly in any of $n-i$ (i=1 . . . $n-1$, where n is total number of dimensions) dimensions. For each cluster some percentage of the cluster points are added as noise to that cluster. There could be every chance that a noise point assigned to a cluster may fall within the bounding box of some other cluster. The major drawback is that the algorithm is limiting the number of dimensions as well as the number of clusters. "Clusutils" [13], has a component called "clusgen"which is a tool based on Glenn's [11] algorithm. Though this tool does not limit the number of dimensions and the number of clusters, it can generate only rectangular/square shaped (a fixed shape) clusters but not random shaped clusters. On the otherhand, SynDECA is capable of generating clusters having different shapes such as circular, elliptical, rectangular, squared, random shape. SynDECA can also generate clusters whose bounding boxes may overlap when k ($< d$, total number of dimensions) dimensions are considered.

[2]The term bounding box is used extensively in this paper to mean minimum bounding box

3. SYNDECA - FRAMEWORK

SynDECA currently deals with numerical dataset generation. In the following sections, we describe the various components of SynDECA and their functionalities.

3.1 Notation

Let the dataset to be generated be represented as X. Let the dimensionality of the dataset be d, the dimensional space be $D \subseteq R^d$ and the number of points in the dataset be n. Let the range of each axis be $[0, m]$, where m is, the maximum allowable value in each dimension. Let c be the number of clusters to be present in the dataset X, X_c be the set of points that belong to clusters and X_n be the set of points that are noise. Let η be the noise percentage, i.e. $\frac{|X_n|}{|X|} \times 100$. Let μ represent the set of cluster centers, μ_i represents the cluster center of cluster i and μ_{ij} is the value of j-th dimension of cluster center i. With the above notation, we define the problem of synthetic numerical cluster dataset generation as:

Given the number of points to be present in the dataset n, the dimensionality of the dataset d, the maximum range of each dimension m and the number of non-overlapping clusters to be present in the dataset c, our aim is to generate a cluster dataset that is spread across all the d dimensions with exactly c ($< n$) non-overlapping clusters and η percentage of noise points within the dataset.

We list out the tasks, step-by-step, that together address the above problem statement. We divide the problem into 4 smaller tasks (as shown in Figure 1):

1. *Cluster placement:* For all the c clusters, the radius of the cluster and the cluster centers are to be determined such that there is no overlapping between the clusters in all d dimensions. The algorithm is in table 1.

2. *Cluster and Noise Cardinality estimation:* For each cluster, the number of points to be placed is in proportion with its radius. The number of points to be placed as noise also needs to be determined.

3. *Filling clusters with points:* In this technique, with the cluster centers and their radii in place, we sprinkle randomly generated points in bounding boxes of various

clusters, till the required number of cluster points $|X_c|$ is achieved (as described in table 2).

4. *Sprinkling noise:* After filling up the clusters with random points, noise points need to be added such that they do not lie within the region of any cluster, in which cluster points are placed. Noise points are generated till the $|X_n|$ number of noise points is achieved (as described in table 3).

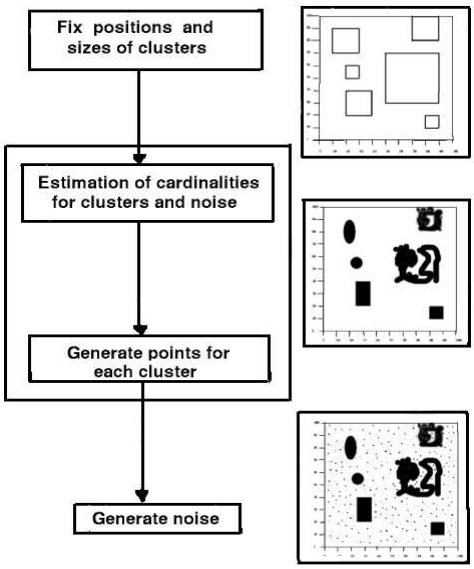

Figure 1: Generation of data

4. ALGORITHMS

With the brief mention of the various tasks in the previous section, we now explain the algorithms in detail. Inputs from the user are: number of points in the dataset n, dimensionality of the dataset d, number of non-overlapping clusters c and maximum allowable range for a dimension m.

Apart from these user-given parameters, we use some more parameters which are set dynamically during the execution of algorithm. They are:

1. *Parameter* ϵ: This parameter is used to ensure a minimum gap between any two clusters, i.e. within $(1+\epsilon)*r$ (r - radius of the cluster) region around the center of a cluster no other cluster can be placed. Its range is (0 , 1] and the value is set randomly.

2. *Maximum radius of a cluster* r_{max}: r_{max} denotes the maximum allowable values for the cluster radius, which is calculated from the user-inputs and ϵ.

3. *Minimum radius of a cluster* r_{min}: This parameter determines the minimum allowable radius for a cluster. This depends on how small a cluster can be when compared with the largest cluster. It is determined by k (ratio of r_{max} to r_{min}. Where k can take any value greater than 1.

4. *Noise percentage* η: η denotes the percentage of noise points. The range of the noise points is $(0\ ,\ \frac{1}{1+c} * n]$. The rest of the points are allocated to each cluster in the proportion to its radius.

4.1 Algorithm for Cluster-Placement

The algorithm proceeds with assigning the center and radius for first cluster randomly. For the remaining clusters, the cluster center and radius are set depending on the placement of the previous clusters. For every cluster, the center μ_i and radius r_i are generated randomly. The generated center is checked such that it is at a distance of radius r_i from the edges along each dimension i.e $m - \mu_{ij} \geq r_i$ or $\mu_{ij} \geq r_i \quad \forall j \in [1, d]$. Once the center is generated, it is checked against each of the already placed clusters for the minimum gap that needs to be maintained. If there is any cluster with which the current cluster does not have the minimum required gap, the radius of the current cluster is reduced. If the reduced radius is less than that of the minimum allowable radius then the process is started again for generating a new center as well as radius. The algorithm is given in Table 1.

Input: $\{(\mu_k, r_k)\}$ Set of the centers and radii of
. already placed clusters (k = 1 . . . i-1),
. r_{min} Minimum allowable radius,
. r_{max} Maximum allowable radius.

Output: μ_i and Radius r_i of the current cluster C_i

Algorithm: Cluster-Placement

1. $r_i \leftarrow$ Generate-radius(r_{min}, r_{max})
/* Generate-radius randomly generates a
value between r_{min} and r_{max} */
2. $\mu_i \leftarrow$ Generate-center(r_i)
/* Generate-center randomly generates a
point in d dimensional space such that
distance between the center and any edge
of bounding box of given space is at least
Radius of that cluster */
3. **while** cluster C_i is not placed
4. **for each** (μ_j, r_j) **in** $\{(\mu_k, r_k)\}$
5. **if** gap between C_i and C_j is not enough **then**
6. Reduce (r_i)
7. **if** ($r_i < r_{min}$) **then**
8. Cluster-Placement$(\{(\mu_k,r_k)\},r_{min},r_{max})$
9. **end if**
10. **end if**
11. **end for**
12. **end while**
13. **end**

Table 1: Algorithm for Cluster-Placement

4.2 Algorithm for Cluster-Points-Generation

After fixing the cluster centers and radii from the Cluster-Placement algorithm, a random number ($\in (0, \frac{1}{1+c} * n]$) of

points are allotted for noise. The remaining points are allotted to each of the cluster in proportion to the cluster's radius. A shape (circle, ellipse, rectangle, square or irregular) for each cluster is assigned randomly. Points for a cluster are generated according to its shape. For a circular shaped cluster, points are generated in such a way that all the points are within a distance of radius r_i from its center μ_i. For rectangular and elliptical shaped clusters, the extent of radius (along a dimension) is reduced in atmost $d-1$ dimensions, points will be within the feasible cluster area. For irregular shaped clusters, we employ two different methods:

1. The first method is analogous to the reverse mechanism of DB-SCAN algorithm [10]. A point p is randomly chosen and a small percentage of the total allocated points (min_pts) are sprinkled around p within a distance of eps ($eps \ll r$). A new point p', which is at a distance of q' ($eps < q' < eps + eps'$ and $eps' \ll r$), from p is chosen and min_pts' points are sprinkled around p' within a distance of eps'. This process is continued till all the required number of points are generated. The condition that the distance q' between points p and p' should be within $(eps, eps + eps')$ ensures the connectivity of the generated points.

2. The second technique is based on the hyper-dimensional grids. In this method, the bounding box of the cluster is divided into small d dimensional hyper cubes, from which one hyper cube is randomly chosen and is filled with points. Following that, an empty hyper cube which is adjacent to the filled hyper cube is chosen to be filled with points. This process is continued till the required number of points are generated for that cluster. In this method, it is ensured that each point that is generated for a cluster is within the bounding box of the cluster.

In case of regular shaped clusters, generated points may follow Gaussian distribution. Where as irregular shaped clusters do not follow any regular distribution. The algorithm is given in table 2. Given a center μ_i and radius r_i, the function *generate-point* generates a point within the bounding box of the cluster. Function *sprinkle-points* generates min_pts points within eps region of a selected point and function *get-a-point* generates a point p' which is at a distance q' from p. Function *is-within-cluster-area* checks whether a point is with-in the feasible area (depending upon the shape of cluster, feasible area changes) of the cluster.

4.3 Algorithm for Noise-Points-Generation

With the cluster points generated, we now generate η percentage of n (number of points in the dataset) noise points. While generating noise points, care is taken such that a point generated for noise will not be with in the space of any cluster where cluster points are generated. The generated point p is a potential noise point if it is not within the bounding box of any cluster. Otherwise if one of the following conditions is satisfied then p is treated as valid noise point. If the shape of the cluster is

- *circle:* p is not within distance r_i from the center of the cluster μ_i.

Input: n_j No.of points to be generated,μ_j Center,
. r_j Radius, Shape of cluster
. $Radii_j$ extent in each dimension for ellipse
. and rectangle, α sprinkle percentage.

Output: n_j points with in the bounding box
. of the cluster

Algorithm: Cluster-Points-Generation

```
1.  if Shape is IRREGULAR
2.    p ← generate-point(μ_j, r_j)
3.    min_pts ← α * n_j
4.    generate min_pts points using
.       sprinkle-points(p,eps,min_pts)
5.   until n_j points are generated do
6.    p' ← get-a-point(p,eps,eps')
7.    min_pts' ← α * n_j
8.    generate min_pts' points using
.       sprinkle-points(p',eps',min_pts')
9.    p ← p'          eps ← eps'
10.  until
11.else
12.   until n_j points are generated do
13.     point ← generate-point (μ_j, r_j)
14.  if  is-within-cluster-area(Shape,point,μ_j,r_j,
.            Radii_j) then
15.                consider 'point' as cluster point
16.     endif
17.   until
18. end if
```

Table 2: Algorithm for Cluster-Points-Generation

- *ellipse:* p is not within the space of the ellipse.
- *rectangle:* for any dimension i, $|p_j - \mu_{ij}|$ is $> Radii_{ij}$ ($Radii_{ij}$ is radius of cluster i along dimension j).
- *irregular:* p is not within the eps distance of any point which is used as pivot to sprinkle the points.

The algorithm is given in Table 3.

4.4 Complexity of Algorithms

SynDECA generates the required datasets within a reasonable time (it does not take more than 10 seconds of time to generate 1,000,000 point, 2-dimensional data containing 100 clusters). For high-dimensional large datasets, it is not easy to evaluate the time analysis because at times, getting the position of a point such that it satisfies various conditions is a time-consuming task. Since the generation of the points and other measures is random, time complexity analysis of the algorithms is beyond the scope of this paper.

5. EVALUATION OF GENERATED DATASETS

Given the user inputs: number of points in the dataset n, dimensionality d, number of clusters c and maximum allowable value in each dimension m, the toolkit generates the required number n points in R^d, d-dimensional space with c

Input: η Percentage of noise points, n Total number of points in the dataset,
. (μ_k, f_k) Set of pairs of center of cluster and other features such as Shape, Radius, Radii of each
. rectangular/elliptical shaped clusters and selected points along with corresponding *eps* in case of irregular
. clusters, m maximum extent of any dimension.

Output: η percentage of noise points.

Algorithm: Noise-Points-Generation
1.**until** η percentage of noise points generated **do**
2. point ← Generate-point-in-space(m)
3. if 'point' is not within the bounding box of any cluster **then**
4. consider 'point' as noise point
5. **else**
6. **if** is-within-free-space-of-cluster(point,(μ_k, f_k)) **then**

/* is-within-free-space-of-cluster function checks whether the point is in the free space of cluster
. and returns true/false accordingly */

7. consider 'point' as noise point
8. **end if**
9. **end if**
10.**end until**
11.**end**

Table 3: Algorithm for Noise-Points-Generation

clusters along with some noise points (calculated based on the other parameter values).

A random number between 0 to $\frac{100}{1+c}$ is selected as noise percent η. The maximum limit on noise percent ensures the average number of points that are given for a cluster is greater than or equal to noise points, which is explained below.

$$\begin{aligned}\left[\frac{100-\eta}{c}\right] * n &\geq \eta * n \\ 100 &\geq (1+c) * \eta \\ \eta &\leq \frac{100}{1+c}\end{aligned}$$

The amount of space provided is, $s = m^d$.

Since any size of bounding box (for c clusters) can not fit in the space provided, there should be a restriction on the size of the bounding box of the cluster. So we need to define a maximum and minimum allowable radius (r_{max}, r_{min}) for each cluster.

Computation of r_{max}:
Maximum space allocated for each cluster is $\frac{s}{c}$. The size of the bounding box of largest cluster (which includes the ϵ space) in terms of r_{max} and ϵ is $[2 * r_{max} * (1+\epsilon)]^d$.

$$\begin{aligned}\frac{s}{c} &= [2 * r_{max} * (1+\epsilon)]^d \\ r_{max} &= \left[\frac{s}{c}\right]^{\frac{1}{d}} * \left[\frac{1}{2 * (1+\epsilon)}\right]\end{aligned}$$

$$\begin{aligned}\text{Minimum allowable radius, } r_{min} &= \frac{1}{k} r_{max} \\ &= \left[\frac{s}{c}\right]^{\frac{1}{d}} * \left[\frac{1}{2 * k * (1+\epsilon)}\right]\end{aligned}$$

where k (> 1), is the ratio of r_{max} to r_{min}. k can take any value, depending on how small a cluster can be when compared to the largest cluster.

Size of bounding box of cluster with r_{max} as radius is

$$\begin{aligned}&= \left[2 * [\frac{s}{c}]^{\frac{1}{d}} * [\frac{1}{2 * (1+\epsilon)}]\right]^d \\ &= \frac{s}{c} * \left[\frac{1}{1+\epsilon}\right]^d\end{aligned}$$

Similarly, the size of bounding box of cluster with r_{min} as radius is

$$= \frac{s}{c} * \left[\frac{1}{k * (1+\epsilon)}\right]^d$$

if every cluster takes r_{max} as its radius, then

$$\begin{aligned}\text{Minimum free space available} &= s - \frac{s}{c}\left[\frac{1}{1+\epsilon}\right]^d * c \\ &= s * \left[\frac{[1+\epsilon]^d - 1}{(1+\epsilon)^d}\right]\end{aligned}$$

Similarly if every cluster takes r_{min} as its radius, then

$$\begin{aligned}\text{Maximum free space available} &= s - \frac{s}{c}\left[\frac{1}{k*(1+\epsilon)}\right]^d * c \\ &= s * \frac{[k*(1+\epsilon)]^d - 1}{[k*(1+\epsilon)]^d}\end{aligned}$$

The step 5 of the algorithm for Cluster-Placement (given in Table 1) checks whether the current cluster i is within the $(1+\epsilon)*r_j$ (radius of cluster j) space around μ_j (center of cluster j, which is already placed). If there is any such overlap then the radius of the current cluster is reduced. Moreover $\epsilon * r_j$ space around the bounding box of cluster j is left free. Hence we can have the following lemma.

LEMMA 1. *The algorithm for Cluster Placement (table 1) generates clusters that are non-overlapping.*

THEOREM 1. *The dataset generated by the above step-by-step process mentioned in section 3, will exactly contain c clusters.*

PROOF. From Lemma 1 it is assured that the bounding boxes of the clusters will not overlap. Each cluster has its own private bounding box, within which there will not be any other cluster's points or the noise points. Now if we can prove that the density of the points within the bounding box of each cluster is more than that of the noise, then we can assure that there exists exactly c clusters (i.e. there will be exactly c sets of points which are denser than the surrounding space) in the generated dataset.

Case 1:

When every cluster has r_{max} as radius and noise percentage is $\eta = \frac{100}{1+c}$

$$\begin{aligned}\text{Number of noise points} &= \frac{\eta}{100} * n \\ &= \frac{1}{1+c} * n\end{aligned}$$

$$\begin{aligned}\text{Total number of cluster points} &= \frac{100-\eta}{100} * n \\ &= \left[1 - \frac{1}{1+c}\right] * n \\ &= \frac{c}{1+c} * n\end{aligned}$$

$\because$ all the clusters have same radius

$$\begin{aligned}\text{Number of points for each cluster} &= \frac{c}{(1+c)*c} * n \\ &= \frac{1}{1+c} * n\end{aligned}$$

The number of points allotted for every cluster is the same as that of the noise points.

$$\begin{aligned}\text{bounding box size of a cluster} &= [2 * r_{max}]^d \\ &= \left[2 * [\frac{s}{c}]^{\frac{1}{d}} * \left(\frac{1}{2*(1+\epsilon)}\right)\right]^d \\ &= \frac{s}{c} * \left[\frac{1}{1+\epsilon}\right]^d\end{aligned}$$

$$\begin{aligned}\text{Total free space} &= \text{Total space provided} \\ &- \sum_{i=1}^{d}(\text{Bounding box size of cluster i})\end{aligned}$$

$$\begin{aligned}\text{Total free space} &= s - c * \frac{s}{c} * \left[\frac{1}{1+\epsilon}\right]^d \\ &= s\left[\frac{(1+\epsilon)^d - 1}{(1+\epsilon)^d}\right]\end{aligned}$$

Inorder to have the density of any cluster to be more than that of the density of noise, (since each cluster has the same number of points) the available free space should be greater than that of the size of the bounding box of the cluster.

$$\begin{aligned}s\left[\frac{(1+\epsilon)^d - 1}{(1+\epsilon)^d}\right] &> \frac{s}{c} * \left[\frac{1}{1+\epsilon}\right]^d \\ \because \frac{s}{(1+\epsilon)^d} &> 0 \\ (1+\epsilon)^d - 1 &> \frac{1}{c} \\ (1+\epsilon)^d &> \frac{1+c}{c} \\ \epsilon &> \left[\frac{1+c}{c}\right]^{\frac{1}{d}} - 1\end{aligned}$$

If the above condition is ensured then the amount of free space is greater than that of the size of the bounding box of each cluster. So we make ϵ to take values within the range $([\frac{1+c}{c}]^{\frac{1}{d}} - 1,\ \ 1]$.

Even when every cluster takes r_{min} as radius, the number of points in every cluster is same as number of noise points. In this case the size of the bounding box of cluster is reduced by $\frac{1}{k^d}$ times and the amount of free space is increased. This means that the same number of noise points(as above) are spread over an increased free space and the same number of cluster points (as above) occupy less space. Therefore even in this case, density of any cluster is more than that of noise density.

Case 2:

When x clusters have got r_{min} as radius, the remaining $c-x$ clusters have got r_{max} as radius and $\eta = \frac{100}{1+c}$, $c-x$ clusters will get more number of points(as x clusters are having minimum radius) than in case 1 and the amount of free space is also increased. Therefore, the density of $c-x$ clusters (with r_{max} as radius) is greater than the noise density.

Points allotted for cluster with r_{max} as radius

$$= \left\lceil \frac{k}{k*(c-x)+x} * \frac{c}{1+c} * n \right\rceil$$

Points allotted for cluster with r_{min} as radius

$$= \left\lceil \frac{1}{k*(c-x)+x} * \frac{c}{1+c} * n \right\rceil$$

When $x = 1$ the cluster with r_{min} as radius gets the least share of points, since the denominator is more when compared to all the other cases.

Example 1. *Let $k = 3$, $c = 10$, $n = 10{,}000$*
when $x = 4$,
Points allotted for cluster with r_{min} as radius = 433

when $x = 1$,
Points allotted for cluster with r_{min} as radius = 324 □

$$\begin{aligned}\text{free space is increased by} \quad & (2 * r_{max})^d - (2 * r_{min})^d \\ = \; & (2 * r_{max})^d - (\frac{2}{k} * r_{max})^d \\ = \; & \left[\frac{k^d - 1}{(k * (1+\epsilon))^d}\right] * \frac{s}{c}\end{aligned}$$

$$\text{Total free space} \quad = \quad \left[\frac{(1+\epsilon)^d - 1}{(1+\epsilon)^d}\right] * s + \left[\frac{k^d - 1}{(k * (1+\epsilon))^d}\right] * \frac{s}{c}$$

The size of the bounding box of the cluster with r_{min} as radius is

$$= \quad \frac{s}{c} * \left[\frac{1}{k * (1+\epsilon)}\right]^d$$

and the density of the cluster with r_{min} as radius is

$$= \quad \frac{\frac{1}{k*(c-1)+1} * \frac{c}{1+c} * n}{\frac{s}{c} * \left[\frac{1}{k*(1+\epsilon)}\right]^d}$$

The density of noise is determined by

$$= \quad \frac{\frac{n}{1+c}}{\left[\frac{(1+\epsilon)^d - 1}{(1+\epsilon)^d}\right] * s + \left[\frac{k^d - 1}{(k*(1+\epsilon))^d}\right] * \frac{s}{c}}$$

$\because$ density of cluster should be greater than noise density.

$$\frac{\frac{1}{k*(c-1)+1} * \frac{c}{1+c} * n}{\frac{s}{c} * \left[\frac{1}{k*(1+\epsilon)}\right]^d} > \frac{\frac{n}{1+c}}{\left[\frac{(1+\epsilon)^d - 1}{(1+\epsilon)^d}\right] * s + \left[\frac{k^d - 1}{(k*(1+\epsilon))^d}\right] * \frac{s}{c}}$$

$$\frac{c * n * c * k^d * (1+\epsilon)^d}{s(1+c)(k * [c-1]+1)} > \frac{n * c * k^d * (1+\epsilon)^d}{s(1+c)[ck^d((1+\epsilon)^d - 1) + (k^d - 1)]}$$

$\because \frac{n*c*k^d(1+\epsilon)^d}{s(1+c)} > 0$

$$\begin{aligned} c \quad & > \frac{k * (c-1) + 1}{c * k^d * ((1+\epsilon)^d - 1) + (k^d - 1)} \\ c^2 * k^d((1+\epsilon)^d - 1) \quad & > kc - k + 1 - c * (k^d - 1) \\ c^2 * k^d((1+\epsilon)^d - 1) \quad & > 1 - k - c(k^d - k - 1) \\ (1+\epsilon)^d \quad & > 1 + \frac{1 - k - c(k^d - k - 1)}{c^2 * k^d} \\ \epsilon \quad & > \left[\frac{1 - k - c(k^d - k - 1)}{c^2 * k^d}\right]^{\frac{1}{d}} - 1 \end{aligned}$$

$$\because \left(\frac{1 - k - c(k^d - k - 1)}{c^2 * k^d}\right)^{\frac{1}{d}} < 1$$

The right hand side of the equation is always negative and as ϵ is always positive, the above condition is satisfied. Therefore, for each cluster the density is more than that of noise. Hence, the theorem is proved. □

5.1 Discussion

Let m_i be the maximum allowable value in each dimension. When $max_{i=1}^d(m_i) \gg min_{i=1}^d(m_i)$, the given space is skewed towards rectangular shape. The bounding box of cluster can no longer be of square shaped. Since r_{max} will be more than that of $min_{i=1}^d(m_i)$. So in this case we should consider the rectangular shaped bounding boxes to generate points.
Let l_i be the length of the side of bounding box of large cluster along dimension i. Let $\frac{m_i}{l_i} = \rho$ i = 1 ... d.
When all the bounding boxes of clusters are large,

$$\begin{aligned}\text{space occupied by each cluster} \quad & = \frac{s}{c} \\ & = \prod_{i=1}^{d} l_i * (1+\epsilon)^d\end{aligned}$$

Where $(\epsilon * l_i/2)$ is the amount of space left free around the bounding box of cluster along dimension i.

$$\begin{aligned} \frac{s}{c} \quad & = \quad \prod_{i=1}^{d} l_i * (1+\epsilon)^d \\ \because \; s \quad & = \quad \prod_{i=1}^{d} m_i \\ \frac{\prod_{i=1}^{d} m_i}{c} \quad & = \quad \prod_{i=1}^{d} l_i * (1+\epsilon)^d \\ \frac{\prod_{i=1}^{d} m_i}{\prod_{i=1}^{d} l_i} \quad & = \quad c * (1+\epsilon)^d \\ \rho^d \quad & = \quad c * (1+\epsilon)^d \end{aligned}$$

Value of either ρ or ϵ needs to be fixed, inorder to get the size of the large bounding box (to get the density of the cluster with largest bounding box). If ρ is fixed then there could be every chance that for some values of c and d, ϵ could be negligible (the gap between clusters will also be negligible). In this scenario we cannot ensure non-overlapping clusters. $\epsilon(< 1)$ is fixed to get the value of ρ with the help of c and d. We can prove that there are exactly c clusters in the generated data in the similar way mentioned in above theorem.

6. EXPERIMENTAL RESULTS

SynDECA[15] is developed in C++ language on Linux platform. For the high dimensional data space, not all the generated points (those will fall within the bounding box of the cluster) will be with in the feasible region of the circular, elliptical or random shaped clusters. As the number of dimensions increases, the ratio of feasible region to bounding box size of the cluster will decrease. In this scenario time taken to generate cluster points will increase. To overcome this, care is taken to make sure that each generated point will be within the feasible region of the cluster.

In the case of the grid method for random shaped clusters, as the number of the dimensions increases, number of cells in the grid will explode. Moreover, there is no control over the size of the bounding box of the cluster. Since the maximum allowable size of each dimension can take a large value, it is not possible to maintain the information about each and

every cell. Since the number of the points allotted to a cluster can be large and a very small percentage of the points are filled in a cell, maintaining information about filled cells can become cumbersome.

Few examples of two dimensional data are given in Figure 2. The time taken to generate datasets for various inputs is given in Table 4. Few three dimensional datasets without noise are shown in Figure 3

No.of Points	No.of dim's	No.of Clusters	Max Value for a dim	Time taken (in sec)
1,000	2	10	100	0.30
10,000	2	10	100	0.10
100,000	2	50	100	0.99
1,000,000	2	50	100	8.22
1,000,000	2	100	100	8.25
1,000,000	2	100	1,000	8.93
10,000,000	2	100	1,000	88.21
100,000	10	100	100	8.55
100,000	10	100	1,000	7.42
1,000,000	10	100	1,000	47.06
100,000	50	50	100	397.92
100,000	50	100	100	500.54
100,000	100	50	100	1,196.1
100,000	100	50	1,000	1,568.31

Table 4: Time taken for generating datasets for various inputs

6.1 Comparison with "clusutils"

Since the current version of the "clusutils" tool is not working properly, we could not give much of the comparison of the output. The figure 4 shows results by "clusutils" in the top row and results by "SynDECA" in the bottom row. The major inputs for clusutils are as follows.

- Number of clusters.
- Number of points.
- Number of dimensions.
- Density level.
- Random noise (this value will be added to each and every dimension of every point).
- Percentage of outliers (this is the percentage of number of extra points to be added as outliers/noise).

Inputs common to both the tools are as follows.

Number of points = 500
Number of clusters = 5
Number of dimensions = 2

Input for the density level to clusutils parameter is 1, means all the clusters will have same density. The first two columns are the datasets without noise points, SynDECA is modified sightly for generating noise less datasets. In the last column the dataset of the clusutils is given random noise as 50.0 and noise points are generated in the case of SynDECA. Even when the inputs are same clusutils have taken different ranges in both x and y dimensions. In the case of SynDECA the first image is in 10 x 10 space and the second image is in 100 x 100 space (Remember here the user can specify the maximum value taken by a dimension). The last image of clusutils shows that there are no clusters even though it is specified that there should be five clusters in the dataset. Whereas SynDECA ensures that there will be required number of clusters in the generated dataset.

7. CONCLUSIONS

The problem of generating synthetic datasets requires a thorough understanding of real-life data and the models that best-fit them. We presented a first-cut solution with theoretical evaluation, to generate clusters of complex shapes in high dimensional data along with the inclusion of noise points. We also showed that SynDECA is superior to the existing dataset generating tool and can generate very high-dimensional datasets with inclusion of noise.

Currently, SynDECA only handles numerical datasets. In future we will address the problem of generating datasets dealing with categorical and heterogeneous data types. Future work also includes handling the rectangular space and other new methods to generate the clusters. In this paper, the algorithms generate clusters which are non-overlapping in all the dimensions. We would like to extend this and consider the case when the clusters overlap with each other along each dimension, yet they are separable only when all the dimensions are taken into consideration (as shown in figure 5, where the two clusters are overlapping when considered only along x-axis and only along y-axis, though they are well-separated).

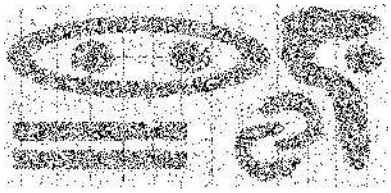

Figure 5: Clusters overlapped in single dimension

8. ACKNOWLEDGEMENTS

We are very much thankful to Kamal Karlapalem for his insightful comments and to Satyanarayana R Valluri, Ravindranath Jampani for proof reading the paper. We also acknowledge the testing of the tool done by Krishnaveni Chitrapu, Final Year B.Tech student at Jaypee Institute of Information Technology, Noida, India, as part of summer internship at International Institute of Information Technology, Hyderabad, India.

9. REFERENCES

[1] D.Fisher. Improving inference through conceptual clustering. AAAI, 1987.

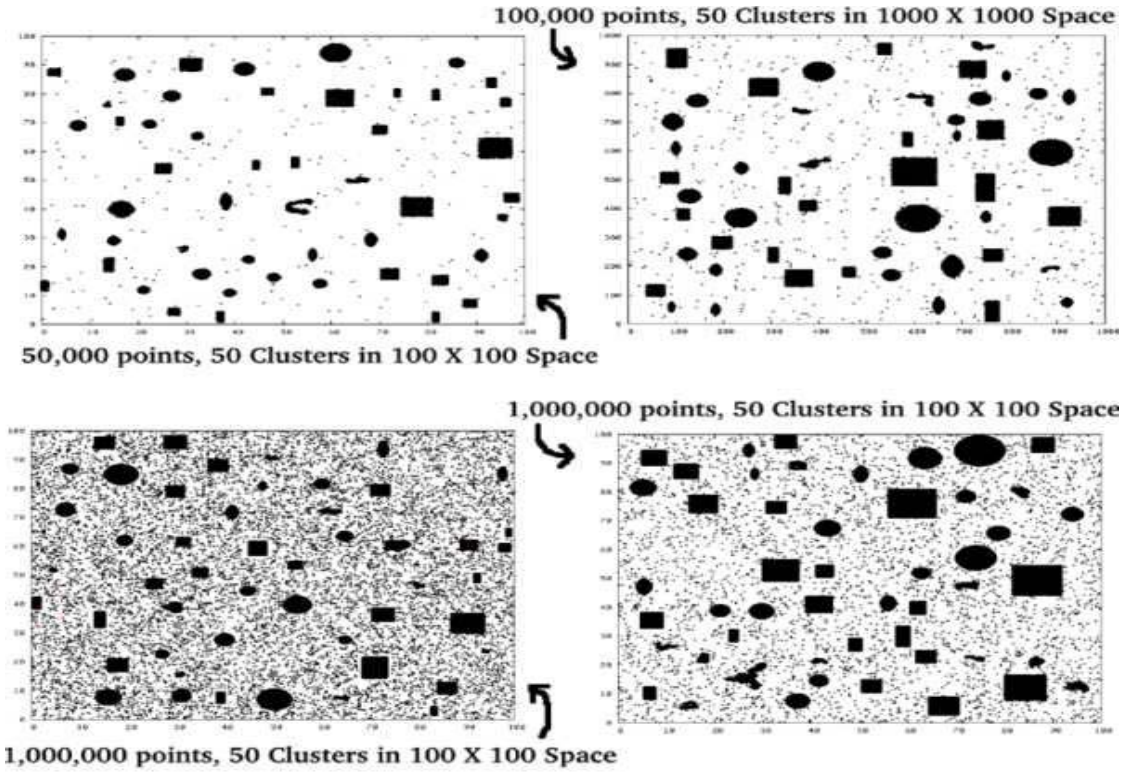

Figure 2: Few examples of the generated data in 2-Dimensions.

[2] J. Edvardsson. A survey on automatic test data generation. ECSEL, pages 21–28, October 1999.

[3] L. Ertoz, M. Steinbach, and V. Kumar. A new shared nearest neighbor clustering algorithm and its applications. Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining, 2002.

[4] L. Ertoz, M. Steinbach, and V. Kumar. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. SIAM International Conference on Data Mining, 2003.

[5] A. Foss and O. R. Zaane. A parameterless method for efficiently discovering clusters of arbitrary shape in large datasets. Proc. of the IEEE International Conference on Data Mining (ICDM'2002), pages pp 179–186, Maebashi City, Japan, December 9 - 12, 2002.

[6] J. Han and M. Kamber. *Data Mining Concepts and Techniques*. Morgan Kauffmann Publishers, 1988.

[7] J.MacQueen. Some methods for classification and analysis of multivariate observations. 5th Berkely Symp. Math. Statist, Prob., pages 281–297, 1967.

[8] L.Kauffman and P.J.Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[9] H. M.Ankerst, M.Breunig and J.Sander. Optics:ordering points to identify the clustering structure. Int. Conf. Management of Data ACM-SIGMOD, 1999.

[10] J. M.Ester, H.P.Kriegel and X.Xu. A density-based algorithm for discovering clusters in large spatial databases. Knowledge Discovery and Data Mining, pages 226–231, 1996.

[11] G. W. Milligan. An algorithm for generating artificial test clusters. Psychometria Vol. 50 No.1 123-127.

[12] M. A. Nascimento, D. Pfoser, and Y. Theodoridis. Synthetic and real spatiotemporal datasets. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2003.

[13] D. X. Pape. Clusutils. Available from: `http://clusutils.sourceforge.net/`, `http://clusutils.sourceforge.net/manual/cg-man.html`, September 2000.

[14] R. S.Guha and K.Shim. Cure: An efficient clustering algorithm for large databases. Int. Conf. Management of Data ACM-SIGMOD, 1998.

[15] SynDECA. http://cde.iiit.net/syndeca/.

[16] R. T.Zhang and M.Livny. Birch: An efficient data clustering method for very large databases. Int. Conf. Management of Data ACM-SIGMOD, 1996.

[17] S. Vadapalli, S. R. Valluri, K. Karlapalem, and P. Gupta. Cluster analysis and outlier detection using reverse nearest neighbors. Technical Report IIIT-H/TR/2004/009, International Institute of Information Technology, Hyderabad.

[18] J. W.Wang and R.Muntz. Sting:a statistical information grid approach to spatial data mining. Int. Conf. Very Large DataBases, 1997.

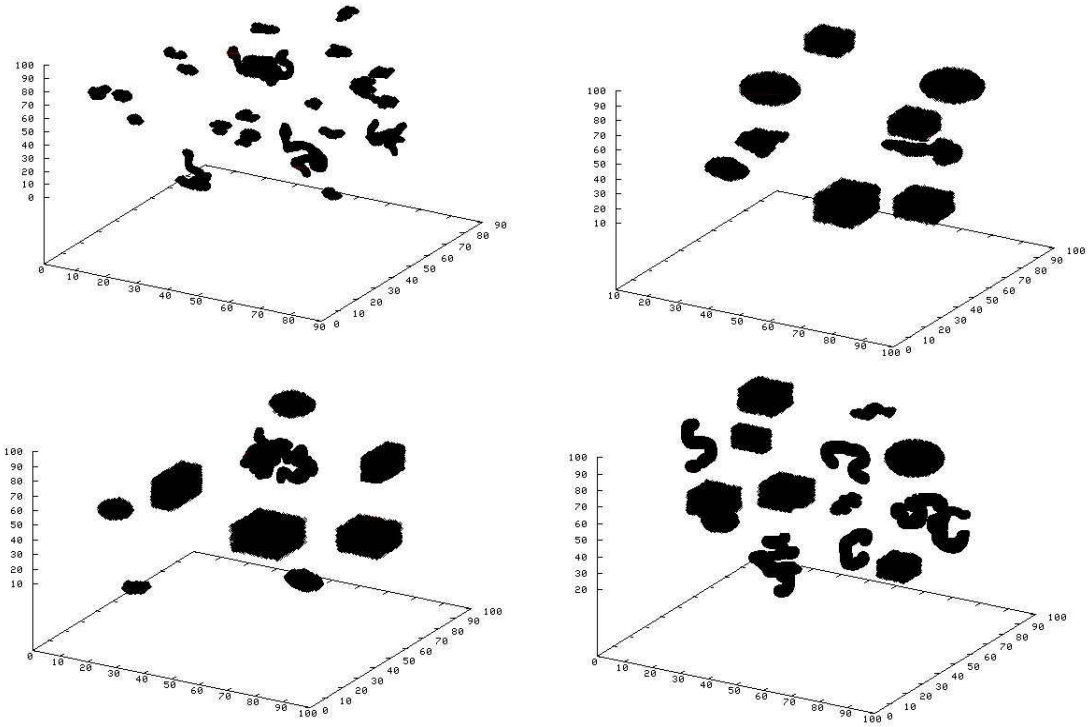

Figure 3: Few 3-D datasets generated without noise.

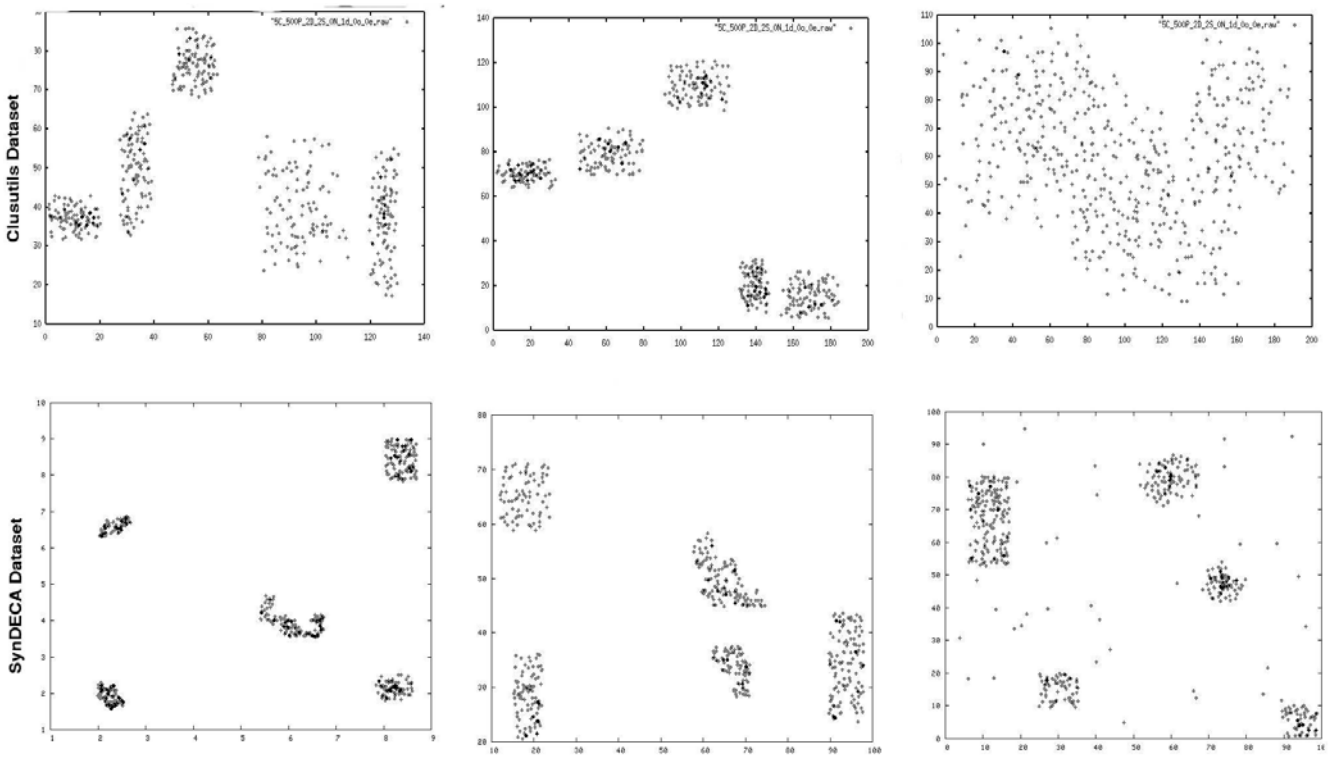

Figure 4: Top row datasets from "clusutils" — Bottom row datasets from"SynDECA".