Journal of Classification 32:241-267 (2015)
DOI: 10.1007/s00357-015-9166-2

TOBAE: A Density-based Agglomerative Clustering Algorithm

Shehzad Khalid

Bahria University, Islamabad, Pakistan

Shahid Razzaq

Bahria University, Islamabad, Pakistan

Abstract: This paper presents a novel density based agglomerative clustering algorithm named TOBAE which is a parameter-less algorithm and automatically filters noise. It finds the appropriate number of clusters while giving a competitive running time. TOBAE works by tracking the cumulative density distribution of the data points on a grid and only requires the original data set as input. The clustering problem is solved by automatically finding the optimal density threshold for the clusters. It is applicable to any *N*-dimensional data set which makes it highly relevant for real world scenarios. The algorithm outperforms state of the art clustering algorithms by the additional feature of automatic noise filtration around clusters. The concept behind the algorithm is explained using the analogy of puddles ('*tobae*'), which the algorithm is inspired from. This paper provides a detailed algorithm for TOBAE along with the complexity analysis for both time and space. We show experimental results against known data sets and show how TOBAE competes with the best algorithms in the field while providing its own set of advantages.

Keywords: Clustering; Agglomerative; Density distribution; Automatic; Noise removal; Non-parametric; Filtering; Terrain; Water puddles; Density threshold.

1. Introduction

An increasing number of systems are now able to capture and store large amount of digital data from all walks of life. General purpose tools are

Corresponding Author's Address: S. Khalid, Department of Computer Engineering, Bahria University, Islamabad, Pakistan 44000, tel: +92-300-5102999, fax: +92-51-2255533, email: shehzad@bahria.edu.pk.

Published online: 3 March 2015

urgently required for mining useful information out of large datasets. This has acted as a spur to the development of content-based data management techniques such as data search and retrieval, discovery of normal patterns, identification of anomalies, classification and prediction. An automated data mining and analysis system is critical to extract useful information from data. Unsupervised learning of patterns is considered to have a pivotal role in higher level data analysis and understanding. A dataset contains in itself different types of semantic patterns each represented by data samples. Normal patterns are quite common and are represented by many samples. Clustering determines these intrinsic patterns, based on similarities between data samples, that are present in datasets by grouping data such that there is low inter-class similarity and high intra-class similarity. The application of clustering includes document clustering (Boley, Gini et al. 1999; Zhao, Karypis, and Fayyad 2005; Tagarelli and Karypis 2013), unsupervised image categorization (Dueck and Frey 2007), grouping and analyzing genes and exons (Frey et al. 2005), constructing treatment portfolios (Dueck, Frey et al. 2008), facility location (Lazic, Frey, and Aarabi 2010; Lazic, Givoni, Frey, and Aarabi 2009) and so on. There exists a variety of techniques that has been used for clustering of data.

1. *Hierarchical Clustering:* Hierarchical techniques combine or divide existing groups according to some similarity measure and maintain a hierarchical structure of the order in which the groups are merged or divided. Agglomerative clustering (Buzan, Sclaroff, and Kollios 2004; Vlachos, Kollios, and Gunopulos 2002; Jain and Dubes 1988, Yager 2000; Everitt, Landau, and Leese 2001) is well known application of hierarchical technique. However, hierarchical approaches are computationally expensive and are not scalable to large datasets. They are also not robust to noise and outliers. Extensions of hierarchical clustering approaches such as BIRCH (Zhang, Ramakrishnan, and Livny 1996), CURE (Guha, Rastogi, and Shim 2001) and ROCK (Guha, Rastogi, and Shim 1999) have been proposed to cater for the drawbacks of hierarchical clustering. BIRCH tackles the problem of scalability and robustness by generating a clustering feature tree storing the summaries of original data. However, BIRCH is only applicable in identification of elliptical clusters with Gaussian densities. Guha et al. proposed CURE and ROCK to cater for the presence of complex shape clusters in the datasets. Both ROCK and CURE employ random sampling to ensure scalability to large datasets. However, ROCK and CURE can not cater for the presence of heterogenous clusters in the datasets. This limitation is handled in CHAMELEON (Karypis, Han, and Kumar 1999). It is an agglomerative clustering

algorithm that operates on K-nearest neighbor graph where the nodes are the data samples and edges are the similarity between data samples. CHAMELEON uses a minimum edge cut to divide the connectivity graph into a large number of relatively small sub-clusters. It then applies agglomerative clustering, based on both relative interconnectivity and relative closeness, to find the genuine clusters by repeatedly combining together these sub-clusters. However, CHAMELEON can not cater for the presence of outliers in training data.

2. *Spectral Clustering:* In recent years, spectral clustering has become one of the popular clustering algorithms having its roots in graph theory. It employs eigenspace decomposition of the symmetric similarity matrix between sample training data. Traditional clustering techniques, such as K-means, are then applied to a subspace of the eigenvectors. Spectral clustering can also identify non-convex clusters such as intertwined spiral. Recently, a number of techniques have been proposed that employ spectral clustering for learning patterns from unclassified data (Ng, Jordan, Weiss et al. 2002; Porikli and Haga 2004; Zelnik-Manor and Perona 2004). Ng et al. (2002) analyze the spectral clustering algorithm using matrix perturbation theory and perform clustering using spectral analysis of affinity matrix computed using samples from a given dataset. However, the proposed algorithm can not automatically identify the number of clusters if such information is not available beforehand. Porilki and Haga (2004) employ spectral clustering for event analysis and can automatically decide on the optimal number of clusters. The proposed algorithm, however, can not deal with clustering data that is distributed according to different scales. Zelnik-Manor and Perona (2004) propose a solution to this problem by calculating a local scaling parameter for each instance instead of having a single scaling parameter for the whole dataset. The scaling parameter is calculated automatically by calculating the distance of the training sample from its 7-NN instance.

3. *Neural Networks:* Neural Networks have been used extensively for clustering (Johnson and Hogg 1996; Owens and Hunter 2000; Stauffer and Grimson 2000; Sumpter and Bulpitt 2000). They have the advantage that a very small number of parameters needs to be optimized for training a network and no a-priori assumptions on the property of data are made. A number of different neural network architectures and methods have been used for unsupervised learning. These include multi-layer perceptrons (Conan-Guez and Rossi 2002), self-organizing maps (SOM) (Owens and Hunter 2000; Khalid

2010b), learning vector quantization (Johnson and Hogg 1996; Stauffer and Grimson 2000; Khalid 2010b; Khalid 2010a; Khalid and Razzaq 2012) etc. Typical neural network based learning algorithms (Johnson and Hogg 1996; Owens and Hunter 2000; Stauffer and Grimson 2000; Kohonen 1997) have a tendency to stuck in the problem of local minima. Khalid (2010b) presented a solution to this problem by initializing the neural network with higher number of output neurons as compared to the desirable number of groupings. Neural network component is then responsible for extracting fine groupings in trajectory data set only once. Hierarchical component uses these fine clusters to generate coarse clusters and, in the process, discovering the actual number of groupings in the trajectory data set.

4. *Partitional Clustering (Square Error based):* In contrast to hierarchical clustering, partitional clustering assigns data samples to K non-hierarchical partitions. K-means (Abraham, Cornillon, Matzner-Løber, and Molinari 2003; Alon, Sclaroff, Kollios, and Pavlovic 2003; Bagnall, Janacek, and Zhang 2003) is a well-known data partitioning method. It is an appropriate technique for learning patterns under the assumption that the clusters are hyperspheroidal. An extension of K-means, referred to as K-Mediods, selects one of the actual data points as a cluster center. Affinity propagation-based approaches have also been proposed recently (Frey and Dueck 2007). Affinity propagation (AP) uses message passing mechanism between training data points to solve the K-Mediod problem by finding representative exemplars within data set with a similarity structure. However, AP requires the specification of two important parameters: preference parameter and the damping factor which is hard to determine. The solution to this problem is provided by Wang, Zhang, Li, Zhang, and Guo (2008). They proposed an adaptive affinity propagation method for clustering to automatically select the preference parameter to identify the correct number of clusters and finding the optimal clustering solution. However, these approaches cannot cater for the presence of anomalies in training data.

5. *Density-based Clustering:* Density-based clustering approaches (Ankerst, Breunig, Kriegel, and Sander 1999; Sander, Ester, Kriegel, and Xu 1998; Hinneburg and Keim 1998; Hinneburg and Gabriel 2007) computes density of samples and uses kernel density estimates to identify dense regions as clusters. DBSCAN (Sander et al. 1998) is the first density based clustering approach that performs density based spatial clustering in input space in the presence of noise. DEN-

CLUE (Hinneburg and Keim 1998; Hinneburg and Gabriel 2007) is a clustering method based on a set of density functions. They use local maximum of Gaussian kernel density function to identify different clusters. A hill climbing approach is used to assign points to different clusters. Although DBSCAN and DENCLUE are efficient clustering approaches with lower computational complexities, they require manual specification of input parameters of cluster radius and density threshold which have significant impact on clustering quality.

From the above discussion, it follows that most of these algorithms break down when the training dataset contains anomalous samples. A major subset of approaches can only learn patterns with elliptical distribution of samples and can not handle complex shape clusters. Most of the approaches require manual specification of parameters such as number of clusters, cluster density threshold etc. that have a significant impact on clustering quality (Karypis, Han, and Kumar 1999; Sander et al. 1998; Hinneburg and Keim 1998; Hinneburg and Gabriel 2007; Abraham et al. 2003; Jain and Dubes 1988; Owens and Hunter 2000). Many unsupervised learning approaches cluster training data by defining the pairwise similarities between training samples (Davies and Bouldin 1979; Ng et al. 2002; Porikli and Haga 2004; Bashir, Khokhar, and Schonfeld 2007a; Bashir, Khokhar, and Schonfeld 2007b; Frey and Dueck 2007; Wang et al. 2008) which is extremely inefficient. The main problem with existing clustering algorithms is that they do not concurrently address all the requirements of clustering such as scalability, handling outliers, learning complex shape clusters, insensitivity to order of input training samples and so on.

Some approaches for clustering in the presence of anomalies have also been presented (Khalid 2010b; Khalid and Naftel 2010; Fraley and Raftery 2002). Khalid and coworkers (Kahalid 2010b; Khalid and Naftel 2010) offer an iterative approach for anomaly detection by clustering the data and iteratively removing clusters affecting cluster quality whilst automatically detecting the number of clusters. However, the proposed approach in Khalid et al. (Khalid 2010b; Khalid and Naftel 2010; Naftel and Khalid 2006) works in the presence of elliptical distribution of clusters. Clustering approach (Stuetzle 2003; Stuetzle and Nugent 2010) based on generalized single-linkage method for hierarchical clustering of data has also been presented. It performs better than other clustering techniques like single-link, average-link, complete link, Wards method and model based clustering - particularly in its ability to handle non-convex clusters. It proposes graph tree pruning over dendrogram cutting, for revealing the final clusters in data. However the choice of the runt excess mass threshold is subjective and a mathematical means of determining it is not given.

In this paper, we present a novel algorithm for learning patterns in the presence of anomalies in training data. The motivations of the proposed learning algorithms are to:

- Develop an unsupervised learning algorithm that efficiently learns the pattern with the space and time complexity much less than $O(n^2)$ where n is the number of training samples.
- Automatically identify the right number of patterns instead of requiring manual information regarding the number and types of groupings hidden in data set.
- Learning in the presence of complex shape distributions of normal patterns
- Minimizing the adverse effects caused by the presence of anomalies in training data, on learning of normal motion patterns.

The remainder of this paper is organized as follows. In Section 2, a novel clustering algorithm, referred to as TOBAE, will be presented. Detailed complexity analysis in spatial and time domain is presented in Section 3. Experiments have been performed to show the effectiveness of proposed system for clustering of complex shape clusters in the presence of anomalies, as compared to competitors. These experiments are reported in Section 4. The last section summarizes the paper.

2. TOBAE: The Proposed Clustering Algorithm

TOBAE algorithm works by capturing the cumulative density distribution of data points in space. Each m-dimensional data point leaves a density signature in the spatial domain and the cumulative effect of these signatures is used in clustering the regions of high density. The algorithm is inspired from the natural phenomenon of water accumulation in puddles on the ground. The same concept can be extended to any m-dimensional scenario. The term '*tobae*' is a Siraiki/Punjabi word, for puddles.

Considering this analogy of puddles, similar to how the ground can be uneven and have depressions, the distribution of data-points is also subject to variations of density. The areas of high data-point density correspond to ground depressions in the puddles analogy, as depicted in Figure 1. Clusters then are simply the regions where puddles of water are formed for a given water level. Using this mechanism, the problem of clustering data breaks down to a single task, that of finding the appropriate water level for the terrain. Different water level scenarios would result in different number and size of puddles on the ground. For non overlapping and spatially separate depressions, the change in water level only effects the size of the

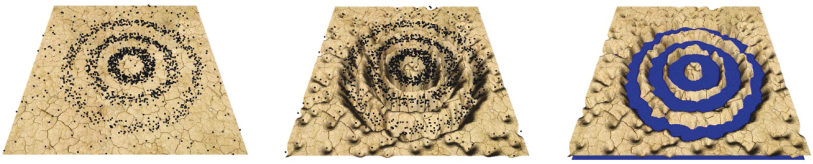


Figure 1. Puddle analogy used to explain data point clustering. Dataset consisting of complex shapes cannot be solved using mean or centroid based approaches. (left) Data points are shown here against the original flat terrain. (center) Areas of high data point density are equivalent to deep depressions in the puddles analogy. (right) The underground water level results in the creation of water puddles.

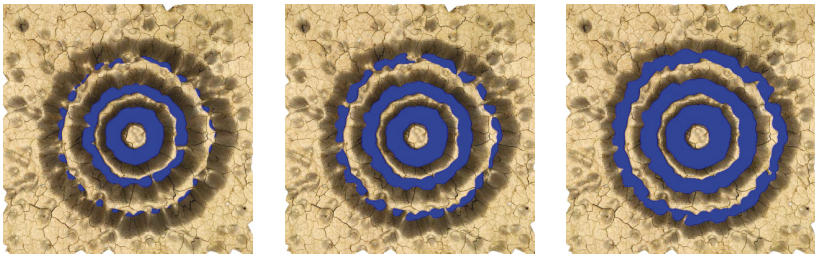


Figure 2. A top view of the terrain with the depth of ground water level decreasing from left to right i.e. going from greater depth to shallow depth. The merging effect of puddles can be seen from left to right and results in the agglomeration of smaller puddles into bigger ones.

water puddle. But for overlapping depressions, this can result in the merger or breaking up of existing puddles. This phenomenon is depicted in Figure 2 and accounts for the agglomerative nature of TOBAE.

The depth of terrain in the puddles analogy is equivalent to the density of data points in the clustering domain. The optimal water level search then is equivalent to finding the optimal density level in the spatial domain of data points. The algorithm is divided into two steps. First is the computation of the cumulative density signatures for the data-points and the second is the optimal density level search. Once the density signatures of the data points have been accumulated in the first step, the algorithm becomes agnostic of individual data-points and all further calculations are based on the density based terrain.

The two high level steps of the algorithm are divided into smaller parts and explained in detail in the following section. These include:

- Terrain Setup

  m-dimensional grid creation

  Spatial indentation computation for data-points
- Optimal density level search

2.1 m-dimensional Grid Creation

The natural phenomenon of *tobae* or puddles takes place in a continuous spatial domain of the real world. In order to simulate that in the discrete domain of digital computers, the spatial domain must be discretized. TOBAE logically divides the spatial domain into equally size m-dimensional hypercubes. The size of these hypercubes is of particular importance to the algorithm and directly effects how accurately TOBAE can generate clusters.

The following scheme is proposed for determining the size of the hypercubes in the TOBAE grid. Let *m* and *n* represent feature size and the number of records/data-points respectively. For a given data set, a histogram of nearest neighbor distances is generated for the data points. The global maximum in this nearest neighbor (NN) distance histogram is used as the size of the grid hypercubes, referred to as μ. For a data set containing n data points, the nearest neighbor distance NN_i (where $1 < i < n$) is calculated for every i^{th} point. A histogram vector H of length b is created for capturing the distribution of nearest neighbor distances with H_0 and H_{b-1} signifying the lowest and highest bins of nearest neighbor distances. The range R_H represented by each bin (or element) of H is given as:

$$R_H = (NN_{max} - NN_{min})/b, \tag{1}$$

where $b = \lceil 3\sqrt{n} \rceil$, NN_{min} and NN_{max} is the smallest and largest nearest neighbor distance value amongst all points in the data set. The number of bins b is kept as a function of the data set size. The size μ of the grid hypercubes can then be calculated as:

$$\mu = NN_{min} + (\imath + 1/2) * R_H, \tag{2}$$

where $\imath$ is the (0 based) index of the global maxima of the histogram and is computed as $\imath = \arg\max_{0 \le i \le b-1} H_i$.

The density based grid is a logical partitioning of the m-dimensional space and is aligned with the axis of the m dimensions. Moreover the indexes of the grid constitute of only positive integer values greater than or equal to 0. The upper bound of the indexes for any dimension is directly related to the highest value of that dimension across all the data points. The input dataset can be represented as a $n \times m$ dimensional matrix D. Let X and Y be two vectors of length *m* elements each, representing the maximum and minimum values of each dimension across the *n* records. These can be written as:

$$X_j = \arg\max_i(D_{i,j}), \tag{3}$$

$$Y_j = \arg\min_i(D_{i,j}), \tag{4}$$

where j ranges from 0 to $m-1$ and i from 0 to $n-1$. For an m-dimensional grid G, the number of cells in the j^{th} dimension, represented as $\aleph_j$, is given as:

$$\aleph_j = ((X_j - Y_j)/\mu) + 1, \tag{5}$$

where j ranges from 0 to $m-1$. Each cell in the grid is like a bin represented by a range of values along a given dimension. The grid computation step is the instance where the nearest neighbor distance measure is used in the proposed algorithm. The order of complexity, with respect to time, for the NN calculations is $O(n^2)$. However this can be reduced using approaches such as early abandonment (Keogh, Wei, Xi, Lee, and Vlachos 2006) and space partitioning trees.

Up to this point, the discussion of the grid has not assumed a particular type of data structure. Owing to the sparse nature of data sets, and consequently the sparsity of the density occupied grid cells, we recommend a hash table based approach which adequately addresses the problem. The algorithm given in this paper and the complexity analysis are both based on a hash table based implementation of the grid. This can be replaced with any other data structure depending on the requirements of time and space.

2.2 Spatial Indentation Computation for Data-points

The previous section sets up a grid for the given dataset. The next step is to add the density signatures of the individual data points to the grid. Once this is complete, all subsequent processing can be performed on the grid and the data points can effectively be discarded.

As can be seen in Figure 1 (center image), each data point leaves a trace of its density signature on the grid. The combination of all such density signatures determines the contours of the terrain. The size and shape of these individual density signatures, or density distribution as it will be called here onwards, is important. We have used the normal (Gaussian) distribution to model the shape of the density distribution for the individual data points. This gives ample weight to nearby cells and lesser weight to ones further away.

The cell size μ for the grid had been based on the nearest neighbor distance criterion and hence the size of the density distribution can now be made as a function of this μ. The size of the Gaussian distribution has a direction correlation with the shape of the cumulative density distribution for the data set. Figure 3 shows three cases of density distribution size for the concentric circles data set. If a very small size is selected, the cumulative density distribution shows disconnected density signatures. A very large size of the density distribution results in the merger of density distribution from

Figure 3. The effects of different sizes of the per-point density signature, on the cumulative density distribution for the data set. (left) Small signature size (center) Correct signature size (right) Large signature size.

across different clusters. The correct size results in the creation of the cumulative density signature that is a good representation of the distribution of data points in space. The standard deviation for the Gaussian density signature is given as $\sigma = C\mu$. The value of constant C is determined empirically and give optimum results with value $C = 3.5$ for all cases.

Using the above mentioned size and shape, the density signature of the n data points of data set D is added to the grid. The general formula for the Gaussian distribution is:

$$f(x) = (1/\sigma\sqrt{2\pi}))e^{(-(x-y)^2/2\sigma^2)}, \tag{6}$$

where y is the mean and x is the variable entity. For the grid, the distance $(x - y)$ is equal to the distance between the cell corresponding to the given data point and the cell for which the density effect is being calculated. If this distance is denoted by d, then the final value of density effect is given as:

$$\widehat{f}(d) = \frac{e^{(-d^2/2\sigma^2)}}{\sigma\sqrt{2\pi}M}, \tag{7}$$

where M is the normalization factor equal to the maximum value of the Gaussian density distribution for the chosen value of σ. The normalized density distribution as a function of μ is shown in Table 1. Around 99% of the density signature is limited within a radius of 3σ or approximately 10μ.

Here is the procedure for computing the cumulative density signature for the data set. Initialize density values of all cells to 0. For each data point in D, do the following:

1. Identify cell I for the data point for which it falls within the cell's corresponding bounding box.
2. For every neighbor cell J within a radius of 10μ from I, calculate Euclidean distance d between centroid of cells J and I.
3. Calculate the density value t for J using this distance d in eq. (7).

Table 1. Normalized density distribution as a function of μ. The density signature for the data points exhibit a Gaussian (normal) distribution.

Distance from density center	0	μ	2μ	3μ	4μ	5μ	6μ	7μ	8μ	9μ	10μ
Density value	1.00	0.96	0.85	0.69	0.52	0.36	0.23	0.14	0.07	0.04	0.02

4. Increment the density value at J by t.
5. Repeat sets 1-4 for all data points.

For an m-dimensional data set, the Gaussian density signature is also m-dimensional and the same eq. (7) is used to calculate the density values for the neighboring cells, using the m-dimensional Euclidean distance d between the cells. The above mentioned steps are repeated for every data point, the result of which is a modified grid which captures the density distribution for all the data points. Using this procedure, the resultant cumulative density distribution for the two-dimensional concentric circles dataset can be seen in Figure 3 (center image). Notice the variations in depth for the depressions throughout the grid, which reasonably capture the density of the original data points. With the density grid populated, the next step is to search for the optimal density level.

2.3 Optimal Density Level Search

Selection of density level is critical in identifying the correct number of clusters. Before the optimal density level search can be explained, it would be useful to draw its parallel in the puddles analogy. Here we introduce the concept of 'ground water level'. The indentations in the grid are analogous to depressions on the ground and are subject to underground water table level. Therefore for a given water level, puddles are formed in regions with depth greater than or equal to the water level, whereas shallower regions do not show any puddles and remain dry. In the puddles analogy, the puddles formed for the optimal 'water level' are the clusters whereas the rest of the depressions are ignored.

Consider the case where the initial water level is equal to the deepest depression in the terrain. In this case only the deepest point(s) in the terrain will form one or more puddles. As the water level is made shallower, the area occupied by the water puddles also increases as seen in Figure 2. At certain values of depths, two or more puddles can merge if the water depth becomes equal to the shallowest point between the two puddles. There are two cases where this puddles merging can happen:

1. Intra-cluster puddle merging: This is the merging of lower density regions of a cluster with those of higher density inside the same cluster.

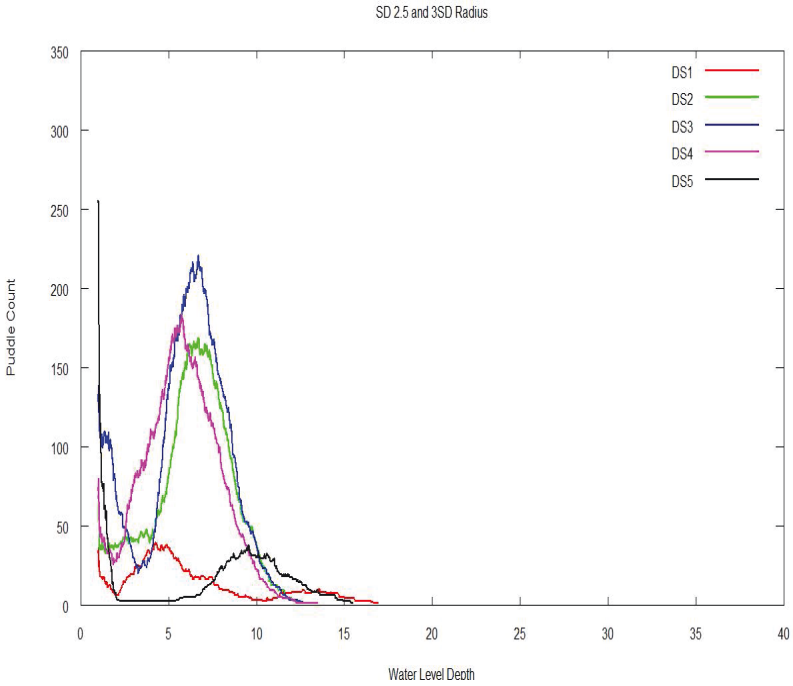


Figure 4. Plots of depth (x-axis) and puddle count (y-axis) for different data sets using $\sigma = 2.5\mu$. For variety of data sets, the same behavior can be observed, i.e. two distinct peaks in the plot with a local minimum in between which represent the optimal water level.

2. Inter-cluster puddle merging: This is the merging of a cluster's puddle with either a neighboring cluster or with puddles corresponding to noise.

Type 1 is desired for clustering whereas type 2 is not. The problem therefore breaks down to that of finding the shallowest water level which would give type 1 puddle merging, without encountering type 2 merging.

Figure 4 shows plot of puddle count vs. water depth for terrain generated for five different data sets. The behavior shown in the plot forms the basis of the optimal density level search. It shows two distinct phenomena. As the water level depth is decreased from the maximum depth value (from right to left inside the plots), there is a sprouting of puddles across the grid, seen as a hike in the puddle count. At some depth the puddle count reaches a local maximum, after which there is a agglomeration or merging stage where the global puddle count decrease due to the type 1 intra-cluster merging of puddles. As the water depth approaches 1, which is the maximum density value in the individual density signature, there is sharp hike in the

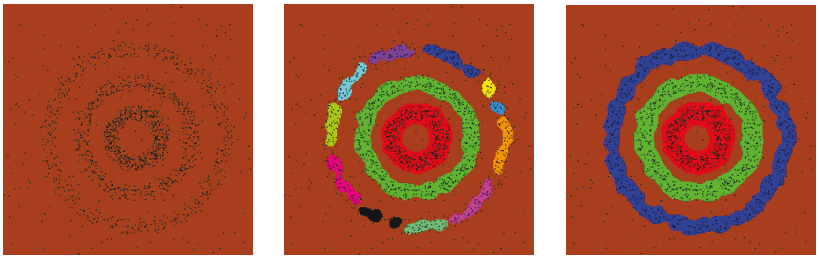


Figure 5. TOBAE algorithm using concentric circles data set. (left) Visualization of the 2D data points. (center) Puddle view for TOBAE for depth similar to the one visualized in Fig. 2 center image. Each color represents a different puddle id. (right) Final puddle view for the data set. Optimal density level is found using the minima in between the two maximas (corresponding to plots of Fig. 4) and noise has been automatically filtered out.

plot. This second hike is attributed to the noise in the data set. Noisy data points are in general spatially distinct and therefore the maximum value of their depth in the terrain is 1 (equal to the maximum value of the normalized density signature). When the water level becomes equal to 1, the depressions corresponding to noise form puddles and result in the puddle count hike. A good candidate for the optimal water depth is one corresponding to the local minimum of puddle count, after the type 1 agglomeration of puddles has occurred. This corresponds to the minimum between the two maxima identified above. This water level will produce very few puddles, thus allowing the constituent parts of individual clusters to merge together, while still avoiding shallower depths where noise is found.

Considering the above mentioned analogy and applying it to the density domain, the density puddles would correspond to connected collections of cells which have density value greater than or equal to the current (global) density level. As density puddles merge with the decrease in this density level, distinct density puddles need to be merged to form bigger puddles. The TOBAE algorithm, as presented later, tracks the individual density puddles using puddle identifiers (pid). Every new density puddle is assigned a new pid. When two or more puddles merge, the result is the discarding of all pids except one, which the new bigger puddle retains. These puddle ids are visually represented as different colors in Figure 5. The puddle analogy is extendable to higher dimensions. Figure 6 shows such a puddle view for a 3-dimensional torus dataset.

The TOBAE algorithm for complex shape clustering, in the presence of anomalies, is presented in Tables 2-6. The algorithm populates the density grid and then finds the optimal density level for the density terrain. Figure 5 shows the visualization of the puddles when running TOBAE.

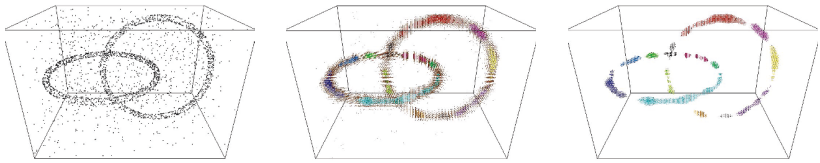


Figure 6. Puddle analogy extended to 3D data sets. (left) Visualization of the 3D data set of two tori with an inter planer angle of 90 degrees. Tori data points have been generated randomly and density of points is not constant along the tori ring. (center) Grid view of all non-zero density cells with brown cells representing region beyond the puddles and the other colors representing puddles. (right) Visualizing of only the puddle cells.

TOBAE uses non contiguous (sparse) grid cells interlinked through two separate data structures. The first is the hash table h for tracking the collection of non-empty cells in the grid g and the second is an array of linked lists a which split the grid cells into buckets of different puddle ids. It is important to note that both the h and a maintain references to the same grid cells i.e. duplicate cells are not created for the data structures. This hybrid data structure is used for efficient space utilization and is discussed in more detail in Section 3.

The clustering of data points is deferred to the clustering of their container grid cells. Initially all grid cells have the default PuddleID of 0. During the course of the algorithm, valid PuddleIDs of 1 and larger can be assigned. As discussed in the previous section, the optimal density level for the grid is found by searching for the minimum puddle count in between the two maximum values. The puddles created at this density level are said to be the clusters for the given data set. Instead of providing the function for finding the minima, we have represented it by the procedure called FindMinimaBetweenTwoMaximas() in the algorithm. Figure 5 (right image) is the puddle view for the concentric circles dataset, found using this optimal density search technique. Notice that the noise in the data set has been automatically filtered.

FindOptimalDensity() is the entry procedure which calculates and tracks the puddle count at different density levels. Puddle count at any time is given by the PuddleIDsUsed variable. The procedure iterates between the density level of maximum density (for the particular terrain) and 1.0 and stores the puddle count after processing of every cell. The density level for this puddle count minimum is the optimal density level and the puddles found at this density level represent the final clusters for TOBAE.

Table 2. Algorithm for entry level procedure in TOBAE

Algorithm *FindOptimalDensity*(p)
Input: A set of data points p
1. Grid g
2. Hashtable h
3. ArrayOfLinkedLists a
4. Initialize(p, g, h)
5. PopulateDensityGrid(g, p, h)
6. LinkandSortRelevantGridCells(h, a)
7. **for** $count \leftarrow 1$ to $a[0].length$
8. $r \leftarrow a[0].RemoveHead()$
9. ClassifyCell(h, a, r)
10. $plot[count].PuddleCount \leftarrow PuddleIDsUsed$
11. $plot[count].Density \leftarrow r.density$
12. $minPuddlesDensity \leftarrow plot.FindMinimaBetweenTwoMaximas()$
13. **return** $minPuddlesDensity$

Table 3. An algorithm for setting up the data structures

Algorithm *Initialize*(p, g, h)
Input: A set of data points p
Input: A grid g
Input: A hashtable h
1. $HashTableSize \leftarrow$ number of data points in p
2. $h.SetSize(HashTableSize)$
3. $g.GridGridParameters(p)$
4. $PuddleIDsUsed \leftarrow 0$
5. $CurrentPuddleID \leftarrow 1$

Table 4. Algorithm to generate the cumulative density signature of the terrain.

Algorithm *PopulateDensityGrid*(g, p, h)
Input: A grid g
Input: A set of data points p
Input: A hashtable h
1. **for** each data point dp in p
2. $c \leftarrow PointCoordToGridCell(g, dp)$
3. **for** each neighbor cell nc of c within r cell radius $dist \leftarrow CellDistance(nc, c)$
4. $val \leftarrow dist * GaussianDistributionValueAtDistance(dist)$
5. **if** $h.ContainsCell(nc) = false$
6. **then** $n.density \leftarrow val$
7. $n.PuddleID \leftarrow 0$
8. $h.add(nc)$
9. **else** $nc.density \leftarrow nc.density + val$
10. $h.update(nc)$

Table 5. Algorithm for filtering grid cells and sorting the remaining in descending order w.r.t. density.

Algorithm *LinkandSortRelevantGridCells*(h, a)
Input: A hashtable h
Input: An Array Of Linked Lists a
1. **for** each cell c in h
2. **if** $c.val <= 1.0$
3. **then** $h.remove(c)$
4. **else** $a[0].add(c)$

Table 6. Algorithm for classifying incoming cell to its corresponding puddle.

Algorithm *ClassifyCell*(h, a, r)
Input: A hashtable h
Input: An Array Of Linked Lists a
Input: A cell r
1. List s
2. **for** each neighbor nc of r present in h
3. **if** $s.ContainsPuddleID(nc.PuddleID) == false$
4. **then** $s.add(nc.PuddleID)$
5. **if** $s.length == 0$
6. **then** $r.PuddleID \leftarrow CurrentPuddleID$
7. $a[CurrentPuddleID].add(r)$
8. $CurrentPuddleID \leftarrow CurrentPuddleID + 1$
9. $PuddleIDsUsed \leftarrow PuddleIDsUsed + 1$
10. **else** $MergePuddleID \leftarrow s.RemoveMin()$
11. $r.PuddleID \leftarrow MergePuddleID$
12. $a[MergePuddleID].add(r)$
13. **while** $s.empty() == false$
14. **do** $puddleToMerge \leftarrow s.RemoveOne()$
15. **while** $a[puddleToMerge].isempty() == false$
16. **do** $t = a[puddleToMerge].RemoveOne()$
17. $t.PuddleID \leftarrow MergePuddleID$
18. $a[MergePuddleID].add(t)$
19. $PuddleIDsUsed := PuddleIDsUsed - 1$

The ClassifyCell() procedure takes the most dense grid cell which has yet to be assigned a PuddleID and adds it to a puddle. Repeating the process for all cells (in descending density, as shown by the procedure call SortLinkedListDescendingByDensity) results in the testing of puddle count at different depths. The final grouping of the individual data points into clusters can thus be achieved by looking at the PuddleID corresponding to their container grid cells at the optimal density level.

In certain cases, adding a new cell to a puddle results in contact between two distinct neighboring puddles (puddles with different PuddleIDs).

This result is the puddle agglomeration caused by the raising of density level. The adjacency criterion for identifying neighbours to merge cells is based on 4-connectivity due to its linear nature.

3. Performance Analysis

Complexity analysis of TOBAE needs to be performed for both time and space. Worst case and average case complexity is analyzed below for both space and time. The input data set contains *n* data points of *m* dimensions each. Each data point creates an *n* dimensional Gaussian density distribution on the grid. The range of this density distribution covers all cells within the distance *r* from the given cell. The upper limit on the number of cells that are processed for inclusion in this distance r is $(2r+1)^m$.

3.1 Time Complexity

The time complexity of TOBAE depends on the time taken to setup the grid and the time taken for processing individual cells for optimal water level search. The setup of grid cells includes two different steps. First is the creation of cells that fall in the range of the density distribution of individual data points and their inclusion in the hash table. Second is the linking of all the cells in the hash table followed by their depth based sorting. Time complexity for the first step in setup is given as:

$$Average\ case : O(n), \tag{8}$$

$$Worst\ case : O(n^2). \tag{9}$$

The time complexity for the second part of setup stage includes linking of the cells in the hash table as well as their sorting. The one time linking of the cells is of order $O(n)$ whereas the subsequent sorting can be performed by any $O(n \log n)$ sorting algorithm. Time complexity of this step is given as:

$$Average\ and\ worst\ case : O(n + n \log n) => O(n \log n). \tag{10}$$

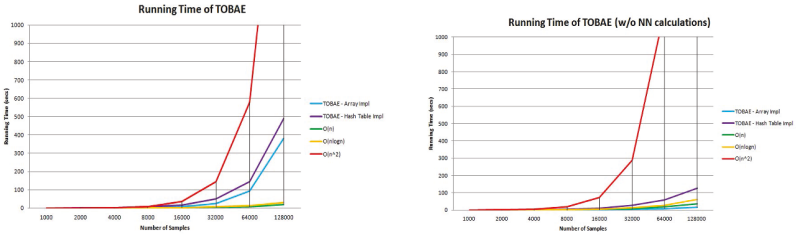
Overall time complexity of setup stage:

$$Average\ case : O(n) + O(n \log n) => O(n \log n), \tag{11}$$

$$Worst\ case : O(n^2) + O(n \log n) => O(n^2). \tag{12}$$

The optimal density level search includes processing each cell in the list one by one and assigning it to a PuddleID bucket. The average and worst case time complexity for the optimal level search is

$$Average\ and\ worst\ case : O(n). \tag{13}$$



(a) (b)

Figure 7. Running time of TOBAE (a) with NN calculation (b) without NN calculations.

Total time complexity of TOBAE, by combining (11), (12) and (13) is:

$$Average\ case : O(n + n\ log\ n) + O(n) => O(n\ log\ n), \tag{14}$$

$$Worst\ case : O(n^2) + O(n) => O(n^2). \tag{15}$$

Figure 7(a) and Figure 7(b) show the running time of TOBAE against a variable number of data points. Figure 7(a) shows the total running time of TOBAE and therefore factors in the NN-distance calculations. The time complexity falls in between $O(n\ log\ n)$ and $O(n^2)$ for both array and hash table based TOBAE implementations.

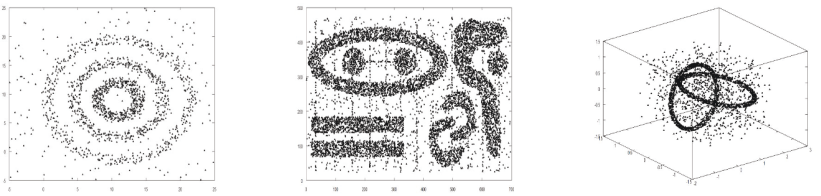
Removing the NN-distance calculations from the running time gives a curve that falls closer to $O(n\ log\ n)$, as seen in Figure 7(b). It is interesting to note that the running time of the array based implementation falls below $O(n)$. This shows that as the number of points is increased in the data set, greater percentage of points falls in previously occupied cells, and therefore does not add a proportional increase in the running time.

3.2 Space Complexity

The space complexity of TOBAE depends primarily on the number of cells used for tracking the density distribution. As the grid can contain a high percentage of unused cells, a hash table based spare grid cell tracking is preferred. Therefore the overall complexity of the TOBAE includes all the used grid cells as well as the memory used for tracking the sparse cells i.e. the hash table memory. Furthermore, array for tracking linked cells is also used for maintaining the PuddleID buckets.

The following terms are used to denote the data structures used by TOBAE:

- g = grid
- h = hash table, contains all occupied cells in g
- a = array of linked lists



DS1 DS2 DS3

Figure 8. Unlabeled training data from DS1-DS3 datasets used in our experiments

Space complexity is therefore given as:

$$O(g + h + a). \tag{16}$$

The grid g is only used to store the properties of the grid and therefore its space is a function of the number of dimensions m. The memory footprint of the hash table includes the table size as well as the actual size of the occupied grid cells. If the table is of length n, then the total space complexity is given as:

$$O(n + n * (2r + 1)^m) => O(n + n * c) => O(n). \tag{17}$$

The maximum size of the PuddleID bucket array 'a' is equal to the number of occupied cells in the grid. Therefore it is given by $O(n * (2r + 1)^m)$. Substituting all the values in (16), the final space complexity of TOBAE is given by:

$$Worst\ case : O(g + h + a) => O(d + n + n * (2r + 1)^d), \tag{18}$$

$$Average\ case : O(n). \tag{19}$$

4. Experimental Results

In this section, we present some results to analyze the performance of the proposed TOBAE algorithm, for complex shape clustering in the presence of anomalies, as compared to competitive techniques. Experiments are conducted on synthetic DS1-DS3 datasets as shown in Figure 8.

4.1 Experiment 1: Comparison of TOBAE with Competitive Techniques

The purpose of this experiment is to compare the performance of proposed TOBAE algorithm with the adaptation of spectral clustering

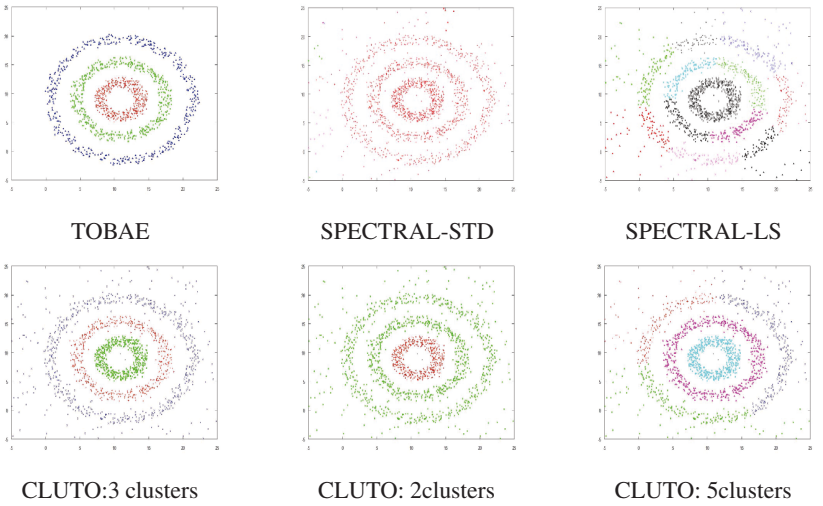


Figure 9. Learning of patterns from DS1 dataset using different clustering algorithm.

(SPECTRAL-STD) (Porikli and Haga 2004; Bashir, Khokhar, and Schonfeld 2007a; Bashir, Khokar, and Schonfeld 2007b), self-tuned spectral clustering with local scaling (Zelnik-Manor and Perona 2004) (SPECTRAL-LS) and CLUTO (Zhao, Karypis, and Fayyad 2005) which is based on CHAMELEON (Karypis, Han, and Kumar 1999). For TOBAE, the σ value of 3.5μ was used. The experiment is conducted on 2-dimensional DS1-DS2 and 3-dimensional DS3 datasets.

Results of learning complex shape clusters using different clusters algorithms for DS1-DS3 datasets are demonstrated graphically in Figures 9-11 respectively. Data points belonging to same class are represented with similar color and marker to ease the visualization of learned clusters. Analyzing results for different datasets shows that TOBAE identifies the right number of clusters for most of the datasets even in the presence of significant number of anomalies. It successfully filters anomalous samples and the overall distribution of normal clusters remains unaffected by the presence of anomalies in training data. Specifically for DS2 dataset where there are dense lines of points connecting different clusters, TOBAE successfully filters them out as anomalous samples. TOBAE automatically performs clustering in the presence of anomalies without the requirement of specifying any manual parameters. On the other hand, quality of SPECTRAL-LS, SPECTRAL-STD and CLUTO is significantly affected by the presence of anomalies. The distribution of normal samples in the identified patterns is also distorted due to the presence of anomalies in the training data. CLUTO also requires

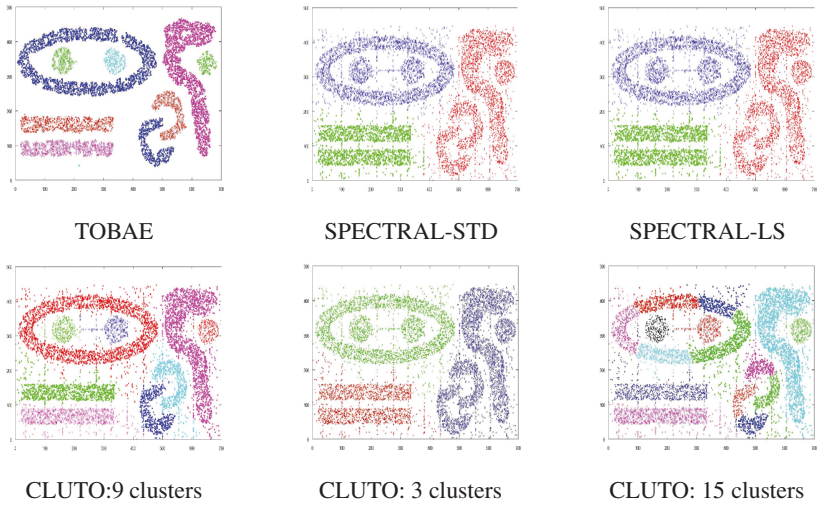


Figure 10. Learning of patterns from DS2 dataset using different clustering algorithm.

manual specification of the number of clusters which is normally not available with unlabeled training data. The quality of CLUTO clustering varies significantly for different number of clusters as highlighted in Figures 9-11.

Comparison of clustering algorithms is now provided by investigating the scalability of these algorithms to the number of samples in the dataset. The response time of clustering algorithms, for different numbers of candidate clusters, is presented in Table 7. We have implemented both array and hashtable based implementation of TOBAE for comparison purposes. It is evident from Table 7 that our proposed approach is an efficient approach that is scalable to large number of samples in the dataset followed by TOBAE. Variants of spectral clustering, on the other hand, are not scalable to the number of samples in the dataset. This unscalability is due to the requirement of computation of an affinity matrix in spectral clustering algorithm. It can also be observed that the array implementation, though having higher memory requirements, is more efficient than the hashtable implementation of TOBAE.

4.2 Experiment 2: Evaluation of TOBAE Using High-Dimensional Data

The purpose of this experiment is to demonstrate the effectiveness of TOBAE in the presence of high dimensional data whilst filtering noisy data. The experiment has been conducted on LAB dataset (Khalid 2010b; Naf-

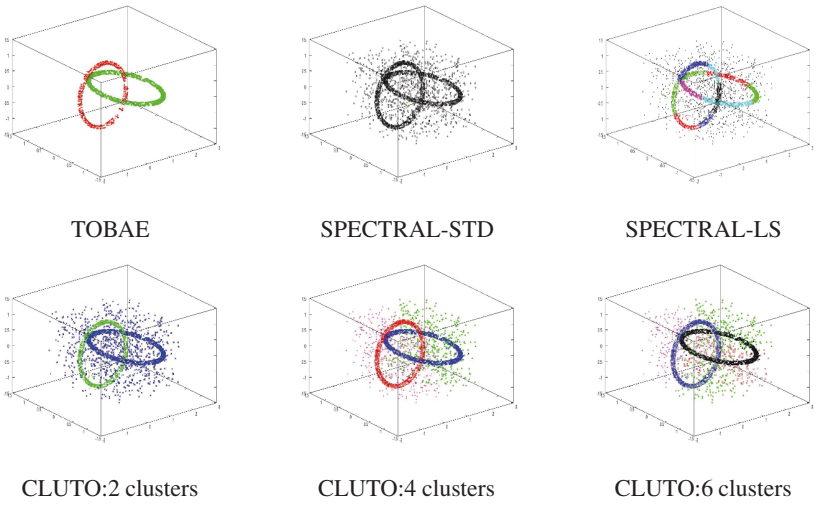


Figure 11. Learning of patterns from DS3 dataset using different clustering algorithm.

Table 7. Comparison of running time of TOBAE with competitors.

	Time taken (secs.)		
	DS1:1703 samples	**DS2:10000 samples**	**DS3:3000 samples**
TOBAE (array impl.)	0.26	2.81	2.17
TOBAE (hash table impl.)	0.37	4.49	13.88
CLUTO	0.97	6.53	1.71
SPECTRAL (local scaling)	67.21	1853	176
SPECTRAL (standard)	65.39	1839	172

tel and Khalid 2006; Khalid and Naftel 2010). LAB dataset is generated by tracking moving objects over a sequence of frames in surveillance environment. The dataset contains trajectories representing four different and distinct motion patterns. Some trajectories, different from the four known motion patterns and representing abnormal behaviour, are also part of LAB dataset. Low-dimensional DFT-MOD based coefficient feature space representation of trajectories are generated as presented in Khalid (2010b). We extracted first 6 Fourier coefficients to generate a compressed feature space representation thus resulting in a 25 dimensional feature vector representation of trajectories. The effectiveness of TOBAE to extract hidden motion patterns, whilst filtering anomalies, from LAB dataset is presented in Figure 12. Figure 12(a) shows complete trajectory dataset superimposed on the background scene. Figure 12(b)-(e) presents the four clusters of trajecto-

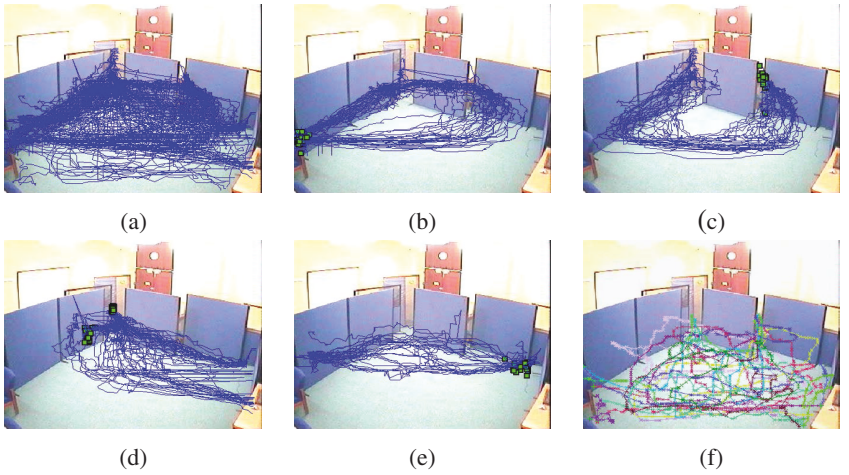


Figure 12. Clustering results of TOBAE using high-dimensional LAB dataset.(a) Complete LAB dataset (b)-(e) Four normal trajectory-based motion patterns identified using TOBAE (f) trajectories filtered as anomalous by TOBAE.

Table 8. Performance of TOBAE in higher dimensional representation of LAB trajectories dataset while keeping the data count constant.

Number of Dimensions	Time Taken (secs.)
17	1.26
25	3.19
41	9.37
57	11.03

ries that have been discovered by our proposed TOBAE algorithm. Squares superimposed on trajectories in the figures depict the starting point of each trajectory. Qualitative inspection shows that trajectories showing similar motion pattern are grouped in same cluster as desired. Trajectories identified as anomalous are presented in Figure 12(f). All the trajectories that do not belong to four planned motion groups are correctly detected as anomalous and are filtered from normal motion patterns.

We further evaluate the performance of TOBAE with respect to time efficiency. We also want to demonstrate the scalability of TOBAE to higher number of dimensions. We generated feature space representation of LAB dataset by having different number of dimensions. This is achieved by using different number of top few Fourier coefficients to represent the trajectories. We employed 4, 6, 10 and 14 top Fourier coefficients thus resulting into 17, 25, 41 and 57 dimensional representation of trajectories. TOBAE tests

were run on a 32-bit 1.86GHz Core(TM)2 machine with 3GB RAM (.NET framework 4.0). It is to be noted that the specified TOBAE implementation capped at 50% CPU usage, and the single-threaded implementation utilized only 1 of 2 cores. The time required by TOBAE to perform clustering of LAB dataset for different number of dimensions is presented in Table 8. We can observe from Table 8 that TOBAE is extremely efficient and it scales well to larger number of dimensions.

5. Conclusion

In this paper, we have presented a novel parameterless density-based clustering algorithm, referred to as TOBAE, that can automatically identify clusters in the dataset whilst filtering noise. The proposed approach computes the cumulative density distribution of the samples in the dataset on a grid. The approach does not require specification of manual density threshold to identify the correct number of clusters. The proposed clustering algorithm is inspired by water accumulation in puddles on the ground, referred to as 'tobae' in Punjabi language. The variation in densities of data samples in a given space is similar to variation in depressions on the ground. For a particular water level, the puddles (clusters) are formed in areas with depth greater than the water level. An optimal selection of water level results in identification of correct number of clusters in the dataset which is determined automatically in our proposed clustering algorithm.

Experimental results are presented to show that proposed TOBAE-based clustering algorithm gives better results than the competitive techniques such as spectral clustering, CURE, DBSCAN and CLUTO. TOBAE based clustering is unaffected by noise in training data and yields correct clustering results. The competitors, on the other hand, are significantly impacted by the presence of noise. We have also shown that the running time of the algorithm is very competitive and tested it against different data sets. TOBAE is one of a kind clustering algorithm as it does not need any supervision, and can therefore be plugged into a pipeline. Debugging of the algorithm is also simple as the N-dimensional problem analysis is broken down to analysis of 2D plots such as 'depth vs. puddle count' and 'nearest neighbor distance histogram'. The TOBAE algorithm is relatively simple and common data structures are used. The space complexity of the algorithm remains $O(n)$ and therefore the low running time is not achieved at the expense of space. Moreover the granularity of the TOBAE grid is customizable, as long as the final signature size (w.r.t μ) for the individual data point remains the same. This makes it scalable to large dimensions as well. TOBAE therefore can be applied to very low end as well as high end machines.

References

ABRAHAM, C., CORNILLON, P.-A., MATZNER-LØBER, E., and MOLINARI, N. (2003), "Unsupervised Curve Clustering Using B-Splines", *Scandinavian Journal of Statistics, 30(3)*, 581–595.

ALON, J., SCLAROFF, S., KOLLIOS, G., and PAVLOVIC, V. (2003), "Discovering Clusters in Motion Time-Series Data", in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1*, 1–375.

ANKERST, M., BREUNIG, M.M., KRIEGEL, H.-P., and SANDER, J. (1999), "Optics: Ordering Points to Identify the Clustering Structure", *ACM Sigmod Record, 28(2)*, 49–60.

BAGNALL, A.J., JANACEK, G.J., and ZHANG, M. (2003), "Clustering Time Series from Mixture Polynomial Models with Discretised Data", University of East Anglia.

BASHIR, F.I., KHOKHAR, A.A., and SCHONFELD, D. (2007a), "Object Trajectory-Based Activity Classification and Recognition Using Hidden Markov Models", *IEEE Transactions on Image Processing, 16(7)*, 1912–1919.

BASHIR, F.I., KHOKHAR, A.A., and SCHONFELD, D. (2007b), "Real-Time Motion Trajectory-Based Indexing and Retrieval of Video Sequences", *IEEE Transactions on Multimedia, 9(1)*, 58–65.

BOLEY, D., GINI, M. et al. (1999), "Partitioning-Based Clustering for Web Document Categorization", *Decision Support Systems, 27(3)*, 329–341.

BUZAN, D., SCLAROFF, S., and KOLLIOS, G. (2004), "Extraction and Clustering of Motion Trajectories in Video", *Proceedings of the 17th IEEE International Conference on Pattern Recognition, 2*, 521–524.

CONAN-GUEZ, B., and ROSSI, F. (2002), "Multi-Layer Perceptrons for Functional Data Analysis: A Projection Based Approach", *Proceedings of the ICANN 2002 Conference on Artificial Neural Networks,* 667–672.

DAVIES, D.L., and BOULDIN, D.W. (1979), "A Cluster Separation Measure", *IEEE Transactions on Pattern Analysis and Machine Intelligence, 2,* 224–227.

DUECK, D., and FREY, B.J. (2007), "Non-Metric Affinity Propagation for Unsupervised Image Categorization, *IEEE 11th International Conference on Computer Vision,* 1–8.

DUECK, D., FREY, B.J. et al. (2008), "Constructing Treatment Portfolios Using Affinity Propagation", *Research in Computational Molecular Biology,* Heidelberg, Germany: Springer, pp. 360–371.

EVERITT, B.S., LANDAU, S., and LEESE, M. (2001), *Cluster Analysis*, London: Arnold, A member of the Hodder Headline Group.

FRALEY, C., and RAFTERY, A.E. (2002), "Model-Based Clustering, Discriminant Analysis, and Density Estimation", *Journal of the American Statistical Association, 97(458),* 611–631.

FREY, B.J., and DUECK, D. (2007), "Clustering by Passing Messages Between Data Points", *Science, 315,* 972–976.

FREY, B.J., MOHAMMAD, N. et al. (2005), "Genome-Wide Analysis of Mouse Transcripts Using Exon Microarrays and Factor Graphs", *Nature Genetics, 37(9)*, 991–996.

GUHA, S., RASTOGI, R., and SHIM, K. (2001), "Cure: An Efficient Clustering Algorithm for Large Databases", *Information Systems, 26(1),* 35–58.

GUHA, S., RASTOGI, R., and SHIM, K. (1999), "ROCK: A Robust Clustering Algorithm for Categorical Attributes", *Proceedings of the 15th IEEE International Conference on Data Engineering*, 512–521.

HINNEBURG, A., and GABRIEL, H.-H. (2007), "Denclue 2.0: Fast Clustering Based on Kernel Density Estimation", in *Advances in Intelligent Data Analysis VII*, eds. M.R. Berthold, J. Shawe-Taylor, and N. Lavra, Springer, pp. 70–80.

HINNEBURG, A., and KEIM, D.A. (1998), "An Efficient Approach to Clustering in Large Multimedia Databases with Noise, *KDD, 98*, 58–65.

JAIN, A.K., and DUBES, R.C. (1988), *Algorithms for Clustering Data*, Upper Saddle River NJ: Prentice-Hall, Inc.

JOHNSON, N., and HOGG, D. (1996), "Learning the Distribution of Object Trajectories for Event Recognition", *Image and Vision Computing, 14(8)*, 609–615.

KARYPIS, G., HAN, E.-H., and KUMAR, V. (1999), "Chameleon: Hierarchical Clustering Using Dynamic Modeling, *Computer, 32(8)*, 68–75.

KEOGH, E., WEI, L., XI, X., LEE, S.-H., and VLACHOS, M. (2006), "LB_Keogh Supports Exact Indexing of Shapes Under Rotation Invariance with Arbitrary Representations and Distance Measures", *Proceedings of the 32nd international Conference on Very Large Data Bases,* 882–893.

KHALID, S. (2010a), "Activity Classification and Anomaly Detection Using¡ i¿ m¡/i¿-mediods Based Modelling of Motion Patterns", *Pattern Recognition, 43(10)*, 3636–3647.

KHALID, S. (2010b), "Motion-Based Behaviour Learning, Profiling and Classification in the Presence of Anomalies", *Pattern Recognition, 43(1)*, 173–186.

KHALID, S., and NAFTEL, A. (2010), "Automatic Motion Learning in the Presence of Anomalies Using Coefficient Feature Space Representation of Trajectories", *Acta Automatica Sinica, 36(5)*, 655–666.

KHALID, S., and RAZZAQ, S. (2012), "Frameworks for Multivariate¡ i¿ m¡/i¿-mediods Based Modeling and Classification in Euclidean and General Feature Spaces", *Pattern Recognition, 45(3),* 1092–1103.

KOHONEN, T. (1997), "Learning Vector Quantization", *Self-Organizing Maps, 30*, 203–217.

LAZIC, N., FREY, B.J., and AARABI, P. (2010), "Solving the Uncapacitated Facility Location Problem Using Message Passing Algorithms", *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 429–436.

LAZIC, N., GIVONI, I., FREY, B., and AARABI, P. (2009), "Floss: Facility Location for Subspace Segmentation", *Proceedings of the IEEE 12th International Conference on Computer Vision,* 825–832.

NAFTEL, A., and KHALID, S. (2006), "Classifying Spatiotemporal Object Trajectories Using Unsupervised Learning in the Coefficient Feature Space", *Multimedia Systems, 12(3),* 227–238.

NG, A.Y., JORDAN, M.I., WEISS, Y. et al. (2002), "On Spectral Clustering: Analysis and an Algorithm", *Advances in Neural Information Processing Systems, 2,* 849–856.

OWENS, J., and HUNTER, A. (2000), "Application of the Self-Organising Map to Trajectory Classification", *Proceedings of the Third IEEE International Workshop on Visual Surveillance,* 77–83.

PORIKLI, F., and HAGA, T. (2004), "Event Detection by Eigenvector Decomposition Using Object and Frame Features, *IEEE Conference on Computer Vision and Pattern Recognition Workshop,* 114–114.

SANDER, J., ESTER, M., KRIEGEL, H.-P., and XU, X. (1998), "Density-Based Clustering in Spatial Databases: The Algorithm Gdbscan and its Applications", *Data Mining and Knowledge Discovery, 2(2)*, 169–194.

STAUFFER, C., and GRIMSON, W.E.L. (2000), "Learning Patterns of Activity Using Real-Time Tracking", *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8),* 747–757.

STUETZLE, W. (2003), "Estimating the Cluster Tree of a Density by Analyzing the Minimal Spanning Tree of a Sample", *Journal of Classification, 20(1),* 025–047.

STUETZLE, W., and NUGENT, R. (2010), "A Generalized Single Linkage Method for Estimating the Cluster Tree of a Density", *Journal of Computational and Graphical Statistics, 19(2)*, 397–418.

SUMPTER, N., and BULPITT, A. (2000), Learning Spatio-Temporal Patterns for Predicting Object Behaviour, *Image and Vision Computing, 18(9)*, 697–704.

TAGARELLI, A. and KARYPIS, G. (2013), "A Segment-Based Approach to Clustering Multi-Topic Documents", *Knowledge and Information Systems, 34(3),* 563–595.

VLACHOS, M., KOLLIOS, G., and GUNOPULOS, D. (2002), "Discovering Similar Multidimensional Trajectories", *Proceedings of the 18th IEEE International Conference on Data Engineering*, 673–684.

WANG, K., ZHANG, J., LI, D., ZHANG, X., and GUO, T. (2008), "Adaptive Affinity Propagation Clustering", *arXiv,* 0805.1096.

YAGER, R.R. (2000), "Intelligent Control of the Hierarchical Agglomerative Clustering Process", *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 30(6),* 835–845.

ZELNIK-MANOR, L., and PERONA, P. (2004) "Self-Tuning Spectral Clustering", *Advances in Neural Information Processing Systems, 17,* 1601–1608.

ZHANG, T., RAMAKRISHNAN, R., and LIVNY, M. (1996), "BIRCH: An Efficient Data Clustering Method for Very Large Databases", *ACM SIGMOD Record, 25(2)*, 103–114.

ZHAO, Y., KARYPIS, G., and FAYYAD, U. (2005), "Hierarchical Clustering Algorithms for Document Datasets", *Data Mining and Knowledge Discovery, 10(2)*, 141–168.