Grid-based Supervised Clustering Algorithm using Greedy and Gradient Descent Methods to Build Clusters

Pornpimol Bungkomkhun[1] and Surapong Auwatanamongkol[2]

**[1] School of Applied Statistics, National Institute of Development Administration
Bangkok 10240, Thailand**

**[2] School of Applied Statistics, National Institute of Development Administration
Bangkok 10240, Thailand**

Abstract

This paper presents a grid-based supervised clustering algorithm being able to identify clusters of any shapes and sizes without presuming any canonical form for data distribution. The algorithm needs no pre-specified number of clusters and is insensitive to the order of the input data objects. The algorithm gradually partitions data space into equal-size grid cells using one dimension at a time. The greedy method is used to determine the order of dimensions for the gradual partitioning that would give the best quality of clustering, while the gradient descent method is used to find the optimal number of intervals for each partitioning. Finally, any connected grid cells containing data from the same class are merged into a cluster. Using the greedy and gradient descent methods, the algorithm can produce high quality clusters while reduce time to find the best partitioning and avoid the memory confinement problem during the process.

Keywords: *Supervised Clustering, Grid-based Clustering, Subspace Clustering, Gradient Descent.*

1. Introduction

Clustering analysis is one of the primary methods to understand the natural grouping (or structure) of data objects in a dataset. The main objective of clustering is to separate data objects into high quality groups (or clusters), based on similarities among the data objects. Due to the acknowledgment that no single clustering method can adequately handle all sorts of cluster structures [1], and that different clustering approaches often define different definitions for clusters, it is impossible to define a universal fitness function to measure clustering quality [13].

Traditional clustering is performed in unsupervised learning manner. No class label attribute of data objects is used to guide clustering them into groups. Since the problem of finding the optimal clustering of data objects was proven to be NP-complete [14], many heuristic methods have been developed to solve the problem. Kotsiantis et al. [8] categorized traditional clustering algorithms into several methods, namely, partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods.

Unlike the goal of traditional unsupervised clustering, the goal of supervised clustering is to identify class-uniform clusters that have high data densities [11],[24]. According to them, not only data attribute variables, but also a class variable, take part in grouping or dividing data objects into clusters in the manner that the class variable is used to supervise the clustering. At the end, each cluster is assigned with specific class label corresponding to the majority class of data objects inside the cluster.

This paper proposes an algorithm that performs supervised clustering based on two methods, i.e. grid-based clustering method and bottom-up subspace clustering method. The proposed algorithm gradually partitions data space into equal-size nonempty grid cells (containing data objects) using one dimension at a time for partitioning and merges the connected grid cells with same data class majorities to form partial clusters until all dimensions have been partitioned. This process follows the framework of bottom-up subspace clustering.

To gradually partition data space into nonempty grid cells, the proposed algorithm first find the order of dimensions to be used for the gradual subspace clustering by considering each of the data dimensions as the only dimension for partitioning (individual dimension subspace clustering). The optimal number of equal intervals to achieve the best quality of supervised clustering for each partitioning of a dimension is determined using gradient descent technique instead of sequential search as in [4]. Next, the bottom up subspace clustering is performed by gradual partitioning data space using one dimension at a time. Greedy approach is used to select next dimension, based on their clustering quality measurement values achieved from the individual dimension subspace clustering, to participate in the grid cell partitioning. The

optimal number of the each partitioning is also determined using the gradient descent method. A dimension is added into the partitioning one at a time until all dimensions have participated in the partitioning. Finally, all connected nonempty cells containing the same majority class of data are merged into the same cluster.

The paper is organized as follows. Section 2 presents backgrounds and related works of the proposed algorithm. Some definitions of the terms used in the proposed algorithm are given in section 3. The detail of the proposed algorithm is described in section 4. Sets of experiments designed to evaluate the effectiveness of the proposed algorithm as well as the experimental results are discussed in section 5. Finally, the conclusions are given in section 6.

2. Background and Related Works

In this section, essential backgrounds on subspace clustering and supervised clustering are provided. Reviews on clustering algorithms relevant to the proposed algorithm are also given.

2.1 Subspace Clustering

Data objects may be related in different ways when different subsets of dimensions are considered. Thus different clusters exist when different sets of dimensions of the data objects are used for clustering [18]. Subspace clustering aims to reveal clusters lying in various subspaces of the dataset. Parsans et al. [13] classified subspace clustering algorithms into two major groups with regard to the search technique employed: the bottom-up method and the top-down method. The bottom-up method searches for clusters from subspaces with smaller subsets of dimensions to subspaces with larger subsets of dimensions, whereas the top-down method does the other way around.

A number of subspace clustering algorithms had been proposed [13]. One of them that the proposed algorithm is based on is CLIQUE [1]. CLIQUE is one of very first subspace clustering algorithms. It is a grid-based clustering algorithm that provides an efficient approach for bottom-up subspace clustering. It uses an APRIORI style technique to find clusters in subspaces, based on the observation that dense areas in a higher-dimensional space imply the existence of dense areas in a lower-dimensional space.

CLIQUE identifies dense clusters in a subspace of maximum dimensionality by automatically identifying projections of the data objects onto various subsets of dimensions where regions of high densities with respect to those objects reside. The algorithm uses a bottom-up approach in generating grid cells and identifying dense cells. It begins by finding dense cells in all one-dimensional spaces, corresponding to each individual attribute of dataset. The algorithm then proceeds level-by-level, in the manner that the candidate *k*-dimensional dense cells can be determined using already determined *(k-1)*-dimensional dense cells. Hence, the set of candidate *k*-dimensional cells that might possibly be dense can be found inside dense *(k-1)*-dimensional cells only. The algorithm terminates when no more candidates are discovered. To form clusters, CLIQUE uses a depth-first search algorithm to find the connected dense cells, then creates cluster descriptions in the form of DNF expression.

2.2 Supervised Clustering

Supervised clustering aims to identify clusters that have high data densities and have minimal impurity, with respect to majority classes of the clusters. The clustering is performed on attribute variables under the supervision of a target class variable. As a consequence, each generated cluster is labeled with only one specific class that has majority of data objects inside the cluster. Supervised clustering procedure is therefore used not only for knowledge discovery, but also for data classification, as the cluster structure with class information can be used as a classification function [21].

Tishby et al. [19], Slonim et al. [17], and Aguilar et al. [2] proposed supervised clustering algorithms based on bottom-up agglomerative approach. The algorithm proposed in [16] is intended to find clusters that are homogenous in the target class variable using a probabilistic approach based on discriminative clustering to minimize distortion within clusters. Qu et al. [20] introduced supervised model-based clustering algorithms that were based on multivariate Gaussian mixture model which employed EM algorithm to estimate model parameters.

Finley et al. [5] proposed that supervised clustering can be achieved by training a clustering algorithm to produce desirable clusters. An SVM algorithm that could learn from an item-pair similarity measure to optimize clustering performance based on a variety of performance measures was proposed. Al-Harbi et al. [3] introduced supervised K-means algorithm that combined Simulated Annealing with K-means algorithm.

CCAS algorithms were developed for detecting intrusions into computer network system through intrusion signature recognition. The algorithms starts by learning data patterns based on supervised clustering procedure, and afterwards use these patterns for data classification. The original version of CCAS [22] starts with two dummy clusters and

allows clusters of each individual class to spread over the entire data space regardless of the training sequence of data objects. Li et al. [9] modified the original CCAS with grid-based method to limit the search space in splitting training data objects into smaller size clusters. The algorithm begins with dividing data space into equal size grid cells. It then performs dummy-based clustering only on data objects lying in the same cell.

Li et al. [10] enhanced the robustness of CCAS by strengthening the algorithm with three post-processing steps: data redistribution, supervised grouping of clusters, and removal of outliers. ECCAS [11] enabled CCAS to handle data of mixed types, by introducing two methods for combining numerical and nominal variables in calculating distance measure. The first method combines different distance measures for each type of variables into a single distance measure ranging between 0 and 1. The second method is based on conversion of nominal variables to binary variables, and then treats these binary variables as numeric variables.

Three representative-based supervised clustering algorithms were introduced in [24]: Supervised Partitioning Around Medoids (SPAM), Single Representative Insertion/Deletion Steepest Decent Hill Climbing with Randomized Start (SRIDHCR), and Supervised Clustering using Evolutionary Computing (SCEC). In their paper, a new fitness function used for measuring the performance of supervised clustering algorithms was proposed. Instead of relying only on the tightness of data objects in each cluster, like most of the traditional clustering algorithms, the three algorithms weights cluster purity against the number of generated clusters in the proposed fitness function.

SPAM, aimed to be the variation of PAM (Partitioning Around Medoids) that uses the proposed fitness function, starts by randomly selecting a medoid from the most frequent class data objects as the first representative. The algorithm then fills up the initial set of representatives with non-representative objects. The number of representatives is fixed by a pre-defined figure. SPAM later on repeatedly explores all possible replacements of a single representative of the most current solution by a single non-representative, provided that the new set of representatives induces minimum fitness function value. The algorithm terminates if none of the replacements can provide lower fitness function value.
In order to eliminate the limitation of SPAM that the number of representatives must be fixed by a pre-defined parameter, SRIDHCR algorithm permits either adding or removing any representatives into or from the current set of cluster representatives. The algorithm terminates when there is no significant improvement in the solution quality (measured by the value of the fitness function).

Besides the above two greedy algorithms, Zeidat et al. [24] also proposed an evolutionary computing algorithm called SCEC. The algorithm evolves a population of solutions, each of which is a set of representatives, over a pre-defined number of generations. The best solution of the last generation is chosen to be the set of representatives for the clustering. Each solution in the initial population is randomly created. Populations of the subsequent generations are generated through three genetic operators: mutation, crossover, and copy. SCEC uses *K*-tournament selection method (with tournament size of $K = 2$) in selecting potential solutions to participate in creating new population. Different adaptive values are used to control the probabilities of applying each of the three genetic operators to generate new solutions for the subsequent generations.

Jirayusakul [6] and Jirayusakul et al. [7] proposed two supervised clustering algorithms based on prototype-based clustering methodology: Supervised Growing Neural Gas (SGNG) and Robust Supervised Growing Neural Gas (RSGNG). The SGNG incorporates Growing Neural Gas network with various techniques such as Type Two Learning Vector Quantization (LVQ2), adaptive learning rates, and cluster repulsion mechanisms. The SGNG also proposed a new validity based on geometry measurement paradigm in order to determine the optimal number of prototypes. Due to drawbacks of the SGNG of being sensitive to the prototype initialization, the sequence of input data objects, and the presence of noises, the RSGNG is intended to be the robust version of SGNG. The RSGNG incorporates SGNG learning schema with the outlier resistant strategy. Moreover, to determine the optimal number of prototypes where data objects may include some outliers, a modified validity index was proposed based on the Minimum Description Length technique.

3. Definitions

The definitions of major terms to be used through out this paper are defined in this section.

3.1 Data Objects

A data object is considered a data point in a *d*-dimensional space. Formally, each data point is a $(d + 1)$-tuple in the form of $\{a_1, a_2, \ldots, a_d, T\}$, where a_i represents value of the i^{th} predictor variable (or attribute) and T represents the value of the target variable (or class label) of the data point [23].

3.2 Grid Cells

Let $A_1, A_2, \dots, A_d$ be sets of dimensions (or attributes, or predictor variables) of any datasets, and let $A = A_1 \times A_2 \times \dots A_d$ be the d-dimensional data space. The problem is to divide the data space A into $\prod_{i=1}^{d} P_i$ non-overlapping hyper-rectangular grid cells, where P_i represents the number of intervals in the i^{th} dimension of d-dimensional data space. A cell is defined by a set of d-dimensional hyperplanes, all of which are parallel to $(d-1)$ coordinate axes.

To accomplish this, the range of the value domain of each dimension A_i is partitioned into P_i number of mutually exclusive equal-size right-opened intervals, $l_i^j = [l_i^j, h_i^j)$, $1 \le j \le P_i$, where l_i^j and h_i^j respectively denotes the start value and end value of the j^{th} interval in the i^{th} dimension, and hence each cell is represented in the form $U = \{I_1, I_2, \dots I_d\}$ [12]. A data object $a = \{a_1, a_2, \dots, a_d\}$, where a_i is the value of the i^{th} dimension, is said to lie in a cell U only if $l_i \le a_i < h_i$ for all I_i.

3.3 Clusters

As defined by [1], a cluster is a maximal set of connected dense cells in d-dimensions. The problem is to separate all identified dense cells D into $D_1, D_2, \dots, D_k$ sets, such that all cells in the set D_i are said to be connected, and no two cells, $U_i \in D_i$, $U_j \in D_j$ with $i \ne j$ are connected. Two d-dimensional cells U_1 and U_2 are declared connected, either in case they share at least one common corner point, or there exists another d-dimensional cell U_s to which both U_1 and U_2 are connected.

If a running number is assigned to each interval in all dimensions, starting from 1 to P_i, where P_i is the number of intervals in the i^{th} dimension, each cell can be represented in the form $U_j = \{I_{1j}, I_{2j}, \dots, I_{dj}\}$, where I_{ij} is the interval number of the cell j in the i^{th} dimension. Cells $U_1 = \{I_{11}, I_{21}, \dots, I_{d1}\}$ and $U_2 = \{I_{12}, I_{22}, \dots, I_{d2}\}$ are claimed connected if all $|I_{i1} - I_{i2}| \le 1$, where I_{i1} and I_{i2} are the interval numbers of the i^{th} dimension of U_1 and U_2 respectively.

3.4 Fitness Function

The objective of supervised clustering is to identify groups of data objects that possess low impurities and few groups as possible. To accomplish this, Zeidat et al. [24] proposed the following fitness function, $q(x)$, as a validity measurement to evaluate the performance of a supervised clustering algorithm,

$$\boldsymbol{q(x) = Impurity(x) + \beta * Penalty(k)}$$

where $\boldsymbol{Impurity(x) = \frac{number\ of\ minority\ objects}{n}}$

$$\boldsymbol{Penalty(k)} = \begin{cases} \sqrt{\frac{k-c}{n}}, & \boldsymbol{k > c} \\ \boldsymbol{0}, & \boldsymbol{k \le c} \end{cases}$$

k = *number of generated clusters*
c = *number of classes*
n = *number of data objects*

where parameter k, c, and n represent the number of generated clusters, the number of classes, and the number of data objects, respectively.

The proposed fitness function consists of 2 contradictory parts, $Impurity()$ and $Penalty()$. Due to the objective of supervised clustering, the $q(x)$ value must be kept as low as possible. Further split of data objects into more clusters may cause a decrease on $Impurity()$ value but an increase in $Penalty()$ value. The parameter β puts a weight on the significance of the $Penalty()$ part against the $Impurity()$ part, i.e. the higher the β value, the higher the significance of the $Penalty()$ part. Normally, the β value is chosen between 0 and 5.

Under the consideration that the above fitness function can certainly lead supervised clustering to yield the most effective solution, this $q(x)$ function is chosen to be the fitness function for the proposed algorithm.

4. Proposed Algorithm

The proposed algorithm is a bottom-up supervised clustering algorithm relying on the combination of the concepts of grid-based clustering and subspace clustering. The basic idea of the proposed algorithm is to create uniform size grid cells over the whole data space, resulting in partitioning data objects into a number of groups abiding by the region of each cell. Clusters are afterward defined by merging together all connected nonempty cells with the same class labels. In consequence, the data objects lying inside the region of such connected cells are claimed to be in the same cluster. For the proposed algorithm, each dimension is partitioned into equal-size intervals, under the condition that the numbers of intervals of different dimensions are allowed to be different. The key to the success of the algorithm is to use the proper number of intervals for each partitioning of a dimension. The number of intervals for each dimension must be carefully selected so the smallest value of the fitness function $q(x)$ is achieved. To fulfill this, the proposed algorithm comprises of 2 steps: Dimension Ordering step and Subspace Clustering step.

4.1 Dimension Ordering Step

This step is aimed to be the preparatory part of the next step, with the objective to re-arrange the order of dimensions to be processed sequentially in the subspace clustering procedure. The step comprises two tasks: clustering based on each individual dimension and dimension sequencing.

4.1.1Clustering Based on Each Individual Dimension

Consider that different sets of data have particular characteristic that might result in non-equivalent competence in the clustering activities. The first task of the Dimension Ordering step is to approximate the clustering potential hidden in each dimension when employed solely in the clustering process. The individual potential for each of a dimension is measured by mean of the value of the possible smallest fitness function, $q(x)$, each specific dimension can produce. To determine the individual potential, a subspace clustering based on each dimension is performed and the fitness function value is evaluated.

With the intention to find the optimal number of intervals (noi) for the dimension partitioning as quickly as possible, the gradient descent method is used in searching for the optimal noi value. During the search, let noi_n represent the current noi value and q_n be the corresponding $q(x)$ value of the noi_n, the new noi value, represented by $noi_{(n+1)}$, is calculated by the following formula.

$$\boldsymbol{noi_{(n+1)} = noi_n - \eta\left[\frac{\Delta q_n}{\Delta noi_n / maxnoi}\right] + momentum}, \text{ if } \boldsymbol{n} > 1$$

$$\boldsymbol{noi_{(0)} = 1 \quad and \quad noi_{(1)}} = 2$$

Where Δq_n represents the difference between the current $q(x)$ value (q_n) and its previous value $\left(q_{(n-1)}\right)$, as well as Δnoi_n represents the difference between the current noi value, noi_n, and it previous value, $noi_{(n-1)}$.

Since the $q(x)$ values are certainly less than 1.0, whereas the noi values are higher than 1, the value of Δnoi_n is normalized by a maximum value of noi, i.e. $maxnoi$, in order to correctly determine the gradient or the ratio between Δq_n and Δnoi_n. The learning rate of the gradient descent formula is represented by the symbol η. The momentum term is employed to give a weigh on the current increment/decrement of the noi in order to avoid the convergence to local optima. The term is formularized as

$$\Upsilon\left(|\Delta \boldsymbol{noi_n}|\right)$$

where Υ is the momentum weight.

The noi value calculated from the formula is always rounded to an integer value. When the noi value gets incremented, the noi value is always rounded up in order to facilitate the forward move during early stages of the search. However, when it is decremented, it is rounded to the nearest integer.

The gradient descent search for the optimal noi value for each individual dimension clustering is terminated when either of the two conditions occurs two times consecutively. We assume when such event happens the optima has been reached, therefore, the search should be terminated. The first condition is that there is no decrement of $q(x)$ value. The second condition is that the absolute value of the gradient value is smaller than a threshold.

In addition, when the search goes backward, i.e. noi is decremented, the current value of noi and the new noi (after decremented from the current noi) will mark the range of noi to be searched. If the search moves current noi value beyond the range, the search is also terminated. This is similar to the movement of a pendulum which can swing only back and forth with gradually reduced range to swing.

4.1.2 Dimension Sequencing Task

Abide by the observation, the dimensions possessing lower $q(x)$ values when working individually have tendency to yield better result when working mutually in the subspace clustering. Using greedy approach, such dimensions should be given higher priority for being used early in the subspace clustering process. Hence, the last task of the Dimension Ordering step is to sort the dimensions into a list in ascending order on their smallest $q(x)$ values achieved from the first task. The list will be used to guide the subspace clustering in the next step.

4.2 Subspace Clustering Step

The intention of this step is to find out the delineation of grid cells that would produce clustering with the possible smallest $q(x)$ value when all dimensions are considered together. The Subspace Clustering step comprises two tasks: Grid Cell Creation task and Cluster Formation task.

4.2.1 Grid Cell Creation Task

In this task, grid cells are created in bottom up fashion by gradually and repeatedly partition data space using one additional dimension at a time. Using the heuristic mentioned in section 4.1.2, the partitioning performs on the dimensions in sequence based on the order list created

by the first step. The partitioning starts from creating the first-level grid cells using the first dimension in the list and the optimal number of intervals for the dimension previously derived in the first step. Only data objects residing in each of the first-level grid cells as well as the grid cell information are then written into the external file, cell by cell. Next, the nonempty first-level grid cells, retrieved from the external file, will be partitioned using the second dimension in the list. Since the grid space has already been partitioned by the first dimension, the optimal number of intervals for the second dimension might be different from the one derived in the first step. Hence, the optimal number of intervals for the second dimension must be derived again using the same gradient descent method as in the first step, except that clusters are formed using the first and the second dimensions, and the number of intervals of the first dimension is fixed. The process then continues on the third dimension and so on until all dimensions have been partitioned.

Refer to the fact that the number of generated grid cells can be computed as $\prod_{i=1}^{d} P_i$ where d represents the number of dimensions and P_i represents the number of intervals in the i^{th} dimensions of d-dimensional space, the number of created cells increases dramatically whenever the number of dimensions increases. When the number of dimensions is large, not all grid cells contain data objects, and the number of grid cells containing data objects is usually tremendously small when compared with the number of created cells.

As only nonempty cells in the current dimensional-level grid space are kept for the processing of the subsequent higher dimensional level. This procedure results in saving a lot of processing time, since large parts of search space are discarded. As a consequence, the proposed algorithm allows only $(d-1)$ dimensional nonempty grid cells to be candidates for the generation of d-dimensional nonempty grid cells.

4.2.2 Cluster Formation Task

To create final clusters (or partial clusters formed during the first or the second step), connected nonempty same-class labeled cells are merged into a same cluster. The input for the cluster formation task is a set of cell blocks D, each of which consists of cell's information, cell's class label, and data objects belonging to that cell. Starting from any cell $U \in D$ as the seed cell to form a cluster, the task searches in D to find for all $U_j \in D$, which are connected with U and have the same class label as U. All U_js are then put into a same cluster as well as removed from D. The task arbitrary selects the next seed cell U from D. It then performs the same process to form the next cluster. The iterative process stops when all cells have been removed from D.

With this cluster formation procedure, the proposed algorithm can generate clusters of any shapes and sizes without presuming any specific mathematical form for data distribution, and can produce identical results regardless of the order in which input data objects are presented.

5. Experimental Results

In this section, the results from two sets of experiments performed to evaluate the effectiveness of the proposed algorithm are discussed in section 5.1 and 5.2. The first experiment was performed on two-dimensional synthetic datasets under the permission of the author of [6], and the second one was performed on datasets obtained from University of California at Irving Machine Learning repository [21].

5.1 The Experiments on 2D Synthetic Datasets

The experiment is intended to affirm the effectiveness of the proposed algorithm under various proclaimed situations through *2-D* geometrical presentations. The proposed algorithm was performed on four 2-D syntactic datasets from [6] representing 4 different scenarios: *Test-1* (1,250 records, 2 classes), *Test-2* (1,000 records, 6 classes), *Test-3* (4,185 records, 4 classes), and *Test-4* (3,421 records, 4 classes). The parameters used in the experiments are as follows:

$$\beta = 0.1,\ \eta = 0.5,\ maxnoi = 100,\ \Upsilon = 0.3$$

The results of the experiments are graphically displayed in Figure 1(a)-1(d). Data objects that are claimed as impurities are encircled with a dark color.

Figure 1(a), illustrates the result from the experiment performed on *Test-1* dataset. It shows two pure cross-board shape clusters: *A* and *B*, one cluster per one individual class. This result confirms that the proposed algorithm has ability to identify any irregular shape clusters.

The result on *Test-2* dataset is shown in Figure 1(b). The proposed algorithm depicts fourteen sparse various shape and density clusters: *A1, A2, A3, B1, B2, C1, C2, C3*, *D1, D2, E1, E2, E3* and *F,* with sparse-and-scattered impurity objects. Two clusters, *A3* and *E3*, contain only one data object each, therefore, they may be counted as outliers.

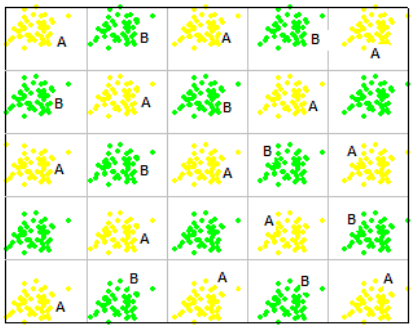

Fig. 1(a) Experimental Result on *Test-1* Dataset.

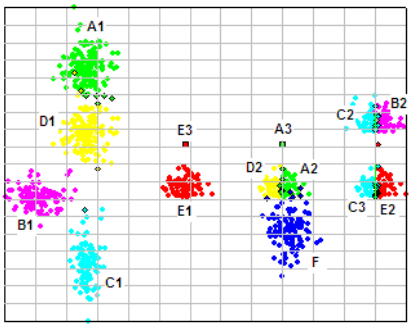

Fig. 1(b) Experimental Result on *Test-2* Dataset.

The set of clusters shown in figure 1(c) is the result from running the proposed algorithm on *Test-3* dataset. Fourteen crowded similar shape, size, and density clusters are delineated: *A1, A2, A3, A4, B1, B2, B3, B4, C1, C2, C3, C4, D1,* and *D2* , most of which are overlapped and contain considerable number of impurity objects.

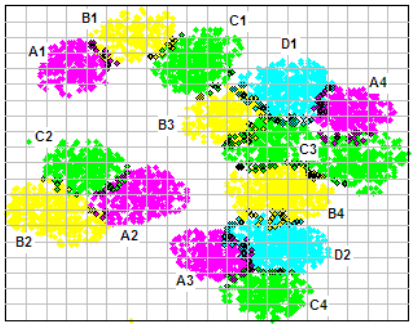

Fig. 1(c) Experimental Result on *Test-3* Dataset.

The results from the experiment performed on *Test-4* datasets are shown in figure 1(d). The proposed algorithm can identify seventeen various size and density clusters: *A1, A2, A3, A4, A5, B1, B2, B3, B4, B5, C1, C2, C3, C4, C5, D1,* and *D2*. Some contain a small number of impurity objects in various locations.

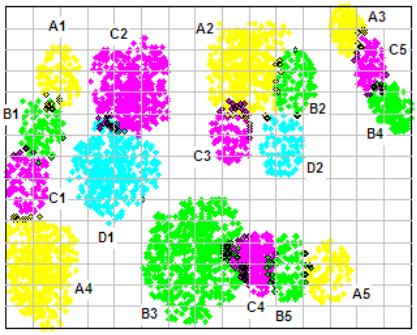

Fig. 1(d) Experimental Result on *Test-4* Dataset.

The results from the four experiments endorse the ability of the proposed algorithm in identifying clusters of any shapes and sizes without presuming any canonical form of data distribution, as well as the ability in handling outliers.

5.2 The Experiments on UCI Datasets

The objective of this set of experiment is to evaluate the performance of the proposed algorithm in a comparative manner with some other supervised clustering algorithms. The experiments were performed on four datasets obtained from University of California at Irving Machine Learning repository [21]: *Iris-Plans* (150 records, 4 attributes, 3 classes), *Pimma-Indian Diabetes* (768 records, 8 attributes, 2 classes), *Vehicle Silhouettes* (846 records, 18 attributes, 4 classes), and *Image-Segmentation* (2100 records, 19 attributes, 7 classes). The results of the fitness values $q(x)$ from the experiments are compared with those results from SPAM, SREDHCR, and SCEC reported in [24], and the best solutions from SGNG and RSGNG in [6]. The parameters used in the experiments are as follows:

$$\beta = 0.1 \text{ and } 0.5,\ \eta = 0.5,\ maxnoi = 100,\ \Upsilon = 0.3$$

Table 1 and table 2 show the experimental results at β value 0.1 and 0.4 respectively. The results in the tables do show that the proposed algorithm yields the best solutions (the smallest $q(x)$ value) among the six algorithms in both values of β. Furthermore, the numbers of clusters generated by the other five algorithms in the last three datasets are remarkably higher than those by the proposed algorithm, due to the nature of representative-based clustering algorithms that incline to create global-shape clusters.

Figure 2 exhibits an example scenario during the search of the optimal *noi* value in one of the experiments. It can be seen that during the forward search of the gradient descent method it can skip a number of *noi* values, e.g. 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 15, 16 and 17, then searches backward

Table 1(a): Experimental Results on ***Iris-Plants*** (*at β=0.1*)

Algorithm	No. of clusters	Cluster purity	*Q(x)* value
SCEC	5	0.993	0.018
SREDHCR	3	0.980	0.020
SPAM	3	0.973	0.027
SGNG	3	0.973	0.027
	5	0.986	0.026
RSGNG	3	0.973	0.027
	5	0.986	0.026
Proposed	***3***	***0.987***	***0.013***

Table 1(b): Experimental Results on ***Pima-Indian Diabetes*** (*at β=0.1*)

Algorithm	No. of clusters	Cluster purity	*Q(x)* value
SCEC	64	0.893	0.135
SREDHCR	45	0.859	0.164
SPAM	45	0.822	0.202
SGNG	45	0.880	0.144
	64	0.919	0.109
	75	0.941	0.090
RSGNG	45	0.863	0.161
	64	0.898	0.130
	75	0.911	0.120
Proposed	***37***	***0.979***	***0.042***

Table 1(c): Experimental Results on ***Vehicle Silhouettes*** (*at β=0.1*)

Algorithm	No. of clusters	Cluster purity	*Q(x)* value
SCEC	***132***	***0.923***	***0.116***
SREDHCR	***65***	***0.835***	***0.192***
SPAM	***65***	***0.764***	***0.263***
SGNG	***65***	***0.861***	***0.166***
	109	***0.920***	***0.115***
	132	***0.946***	***0.093***
RSGNG	***65***	***0.873***	***0.154***
	109	***0.937***	***0.098***
	132	***0.955***	***0.084***
Proposed	***23***	***0.998***	***0.017***

Table 1(d): Experimental Results on ***Image-Segmentation*** (*at β=0.1*)

Algorithm	No. of clusters	Cluster purity	*Q(x)* value
SCEC	***60***	***0.989***	***0.026***
SREDHCR	***53***	***0.980***	***0.035***
SPAM	***53***	***0.944***	***0.071***
SGNG	***42***	***0.967***	***0.046***
	53	***0.971***	***0.044***
	60	***0.977***	***0.039***
RSGNG	***42***	***0.959***	***0.054***
	53	***0.963***	***0.052***
	60	***0.969***	***0.047***
Proposed	***25***	***0.993***	***0.016***

Table 2(a): Experimental Results on ***Iris-Plants*** (*at β=0.4*)

Algorithm	No. of clusters	Cluster Purity	*Q(x)* Value
SCEC	3	0.987	0.013
SREDHCR	3	0.987	0.013
SPAM	3	0.973	0.027
Proposed	***3***	***0.987***	***0.013***

Table 2(b): Experimental Results on ***Pima-Indian Diabetes*** (*at β=0.4*)

Algorithm	No. of clusters	Cluster purity	*Q(x)* value
SCEC	9	0.819	0.219
SREDHCR	2	0.776	0.224
SPAM	2	0.772	0.227
Proposed	***2***	***0.809***	***0.191***

Table 2(c): Experimental Results on ***Vehicle Silhouettes*** (*at β=0.4*)

Algorithm	No. of clusters	Cluster purity	*Q(x)* value
SCEC	61	0.857	0.247
SREDHCR	56	0.835	0.265
SPAM	56	0.754	0.345
Proposed	***5***	***0.996***	***0.017***

Table 2(d): Experimental Results on ***Image-Segmentation*** (*at β=0.4*)

Algorithm	No. of clusters	Cluster purity	*Q(x)* value
Image-Segmentation			
SCEC	28	0.969	0.069
SREDHCR	32	0.970	0.074
SPAM	32	0.940	0.103
Proposed	***12***	***0.984***	***0.036***

to the *noi* value of 16, and finally terminates at the optimal value of 14.

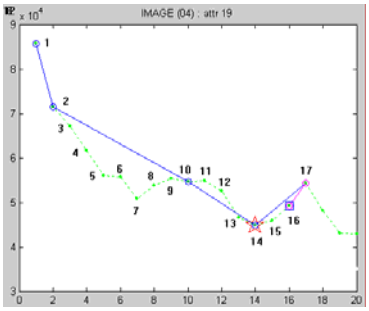

Fig. 2 A Gradient Descent Search for the Optimal *noi* Value.

6. Conclusion

The proposed algorithm possesses all of the good clustering properties mentioned in [15]. The algorithm has ability to produce identical results regardless of the order of data objects to be processed. It can automatically determine the optimal number of clusters and handle clusters of arbitrary shapes and sizes without making any assumption about the distribution of data objects. Moreover, the proposed algorithm possesses the ability to find the optimal number of intervals to be used in partitioning each dimension. The results from the experiments do confirm that the proposed algorithm can cope with datasets of any shapes and sizes. It can outperform some other supervised clustering algorithms on

irregular-shape datasets with smaller numbers of created clusters and lower degrees of impurity.

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," In Proc. ACM SIGMOD International Conference on Management of Data, Seattle, Washington, June 1998, pp. 94-105.

[2] J. S. Aguilar, R. Ruiz, J. C. Riquelme, and R. Giraldez, "SNN: A supervised clustering algorithm," In Proc. 14th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE 2001), Budapest, Hungary: Springer-Verlag, 2001, pp. 207-216.

[3] S. H. Al-Harbi, and V. J. Rayward-Smith, "Adaptive k-means for supervised clustering," Applied Intelligence, Volume 24, Number 3, June 2006, pp. 219-226.

[4] P. Bungkomkhun, and S Auwatanamongkol, "Grid-based Supervised Clustering" In Proc. World Academy of Science, Engineering and Technology 60 2009, Bangkok, Thailand, December 2009, pp. 536-543.

[5] T. Finley and T. Joachims, "Supervised clustering with support vector machines," In Proc. International conference on Machine learning, Bonn, Germany, August 2005, pp. 217-224.

[6] A. Jirayusakul, "Supervised growing neural gas algorithm in clustering analysis," Ph.D. dissertation, School of Applied Statistics, National Institute of Development Administration, Thailand, 2007, unpublised.

[7] A. Jirayusakul, and S. Auwatanamongkol, "A supervised growing neural gas algorithm for cluster analysis," International Journal of Hybrid Intelligent Systems, Vol. 4, No.2, 2007, pp. 217-229.

[8] S. B. Kotsiantis and P. E. Pintelas, "Recent Advances in Clustering: A Brief Survey," Transactions on Information Science and Applications, 2004,1(1), pp.73-81.

[9] X. Li and N. Ye, "Grid-and Dummy-Cluster-Based Learning of Normal and Intrusive Clusters of Computer Intrusion Detection," Journal of Quality and Reliability Engineering International, Vol. 18, No. 3, 2002, pp. 231-242.

[10] X. Li and N. Ye, "A Supervised Clustering Algorithm for Computer Intrusion Detection," Knowledge and Information Systems, Vol. 8, No.4, 2005, pp. 498-509.

[11] X. Li and N. Ye, "A Supervised Clustering and Classification Algorithm for Mining Data With Mixed Variables," IEEE Transactions on Systems, Man, and Cybernetics-Part A, Vol. 36, No. 2, 2006, pp. 396-406.

[12] N. H. Park and W. S. Lee, "Statistical Grid-based Clustering over Data Streams," SIGMODD Record, Vol.33, No.1, March 2004, pp. 32-37.

[13] L. Parsans, E. Haque, and H. Liu, "Subspace Clustering for High Dimensional Data: A Review," ACM SIGKDD Explorations Newsletter, 6(1), June 2004, pp. 90-105.

[14] C. M. Procopiuc, 1997, "Clustering Problems and their Applications (a Survey)," Available: http://www.cs.duke.edu/~magda.

[15] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases," In Proc. International Conference on Very Large Databases, New York City, August 24-27, 1998, pp. 428-439.

[16] J. Sinkkonen, S. Kaski, and J. Nikkila, "Discriminative Clustering: Optimal Contingency Tables by Learning Metrics," In Proc. European Conference on Machine Learning (ECML'02), Springer-Veriag, London, 2002, pp. 418-430.

[17] N. Slonim and N. Tishby, "Agglomerative Information Bottleneck," In Proc. Neural Information Processing Systems, 1999, pp. 617–623.

[18] P-N. Tan, M. Steinbach, and V. Kumar, "Introduction to Data Mining. Boston: Pearson Education, Inc., pp. 487-644.

[19] N. Tishby, F. C. Pereira, and W. Bialek, "The Information Bottleneck Method," In Proceedings of the **37th** Allerton Conference on Communication , Control, and Computing, 1999, pp. 368-377.

[20] Y. Qu and Z. Xu, "Supervised Clustering Analysis for Microarray Data Based on Multivariate Gaussian Mixture," Bioinformatics. 20 (Aug) : 1905-1913.

[21] University of California at Irving, Machine Learning Repository. Available: http:/www.ics.edu/~mlearn/MLRepository.html

[22] N. Ye and X. Li, "A Scalable Clustering Technique for Intrusion Signature Recognition," In Proc. The 2001 IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY, 5-6 June, 2001, pp. 1-4.

[23] N. Ye and X. Li, 2005, "Method for Classifying Data using Clustering and Classification Algorithm Supervised," Available: http://www.patentstorm.us/patents/6907436/fulltext.html.

[24] N. Zeidat, C. F. Eick, and Z. Zhao, "Supervised Clustering: Algorithms and Applications," Available: http://www2.cs.uh.edu/~ceick/kdd/ZEZ06.pdf.

Pornpimol Bungkomkhun is a PhD candidate at the School of Applied Statistics, National Institute of Development Administration, Bangkok, Thailand. She received a Master's degree in Business Computer Information System from North Texas State University, USA, and completed her Bachelor's degree in Statistics / Electronic Data Processing from Chulalongkorn University, Bangkok, Thailand.

Surapong Auwatanamongkol is an associate professor at the School of Applied Statistics, National Institute of Development Administration, Bangkok, Thailand. He received a Ph.D. degree in Computer Science from Southern Methodist University, USA, a Master's degree in Computer Science from Georgia Institute of Technology, USA, and a Bachelor's degree in Electrical Engineering from Chulalongkorn University, Bangkok, Thailand. His fields of research include Pattern Recognition, Data Mining, Evolutionary Computation, Parallel and Distributed Computing. He has published several papers in various international journals and conferences.