# Generalizing the $k$-Windows Clustering Algorithm in Metric Spaces

D.K. Tasoulis [a,b], M.N. Vrahatis [a,b]

[a] *Computational Intelligence Laboratory, Department of Mathematics, University of Patras, GR-26110 Patras, Greece*

[b] *University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR-26110 Patras, Greece*

**Abstract**

Clustering methods are one of the key steps that lead to the transformation of data to knowledge. Clustering algorithms aim at partitioning an initial set of objects into disjoint groups (clusters) such that that objects in the same subset are more similar to each other than objects in different groups. In this paper we present a generalization of the $k$-Windows clustering algorithm in metric spaces. The original algorithm was designed to work on data with numerical values. The proposed generalization does not assume anything about the nature of the data per se, but only considers the definition of a distance function over the dataset. The efficiency of the proposed approach is demonstrated in various datasets.

*Key words:* clustering, data mining, $k$-Windows

## 1 Introduction

An important issue in automated knowledge discovery from data is the partitioning of the data into disjoint and homogeneous meaningful groups, *clusters*. Typical clustering applications include data mining [1], statistical data analysis [2], compression and vector quantization [3], global optimization [4,5] and web personalization [6].

The first comprehensive foundation of clustering methods was published in 1939 [7], but the first references to clustering date back to the fourth century

*Email addresses:* dtas@math.upatras.gr (D.K. Tasoulis), vrahatis@math.upatras.gr (M.N. Vrahatis).

B.C. by Aristotle and Theophrastos and in the 18th century to Linnaeus [8].

The increasing number of database applications in fields such as multimedia content-based retrieval, time series and genome sequencing, involves data items expressed as discrete values without any ordering. The similarity among objects of this kind is measured by a distance function that is defined on the application domain. To this end, clustering algorithms capable of extracting rules in metric spaces need to be developed.

The unsupervised $k$-Windows clustering algorithm [9–11], is a recently proposed algorithm that has been successfully applied in numerous applications including bioinformatics [12,13], medical diagnosis [14,15], time series prediction [16,17] and web personalization [6]. The unsupervised $k$-Windows algorithm has been designed for numerical data lying in a Euclidean $d$–dimensional space. In this contribution, we extend the $k$-Windows algorithm to be directly applicable in metric spaces. The clustering performance of the proposed approach is demonstrated for various datasets. An important feature, of the $k$-Windows algorithm is its ability to provide an aproximation for the number of clusters present in the data. The proposed generalization also provides an efficient implementation of this feature.

The rest of the paper is organized as follows: Section 2 is devoted to the presentation of related work on clustering on metric spaces; Section 3 outlines the workings of the $k$-Windows clustering algorithm on numeric databases. The proposed Generalized Unsupervised $k$-Windows algorithm is presented in Section 4 along with a discussion of its computational complexity. In Section 5 the datasets employed for the evaluation of the algorithm, as well as, the obtained experimental results are presented. The paper ends with conclusions.

## 2 Related Work

Approaches similar to the present one involve algorithms that have the ability to cluster large datasets with mixed categorical and numerical values under arbitrary metrics. Formally, we assume that each object in the database is represented by the same set of attributes, $A_1, A_2, \ldots, A_m$. Each attribute describes the domain of values denoted by $DOM(A_i)$. A domain is defined as categorical if it is finite and unordered. A numeric domain is represented by continuous values. A numeric database contains objects that are described by numeric attributes. A categorical database, on the other hand, contains objects which are described by categorical attributes. Finally, a database is mixed if it contains both numeric and categorical attributes.

Various metrics have been proposed [19,20] that enable standard hierarchical

clustering algorithms to handle mixed databases. However, the non-linear time complexity typically associated with this kind of algorithms restrains their application on large datasets.

Based on the $k$-means procedural scheme, several algorithms that attempt to tackle datasets with mixed attributes have been proposed [21,22]. The main drawback of these approaches is that they require from the user to determine the number of clusters prior to the execution of the algorithm.

More advanced approaches to clustering, like CLARANS [23], are able to provide high quality results, but require a close to quadratic running time [24]. Furthermore, to provide an approximation to the number of clusters, these algorithms require successive re-executions that render the total running time even larger.

Density based approaches like OPTICS [25] and GDBSCAN [24] appear to be the most effective Their running time, however, under the assumption of a spatial indexing, is of the order $O(n \log(n))$, that can also be considered high. A further drawback of these approaches is the high memory requirements during their execution.

## 3 Unsupervised $k$-Windows Algorithm

In this Section, for completeness purposes, we briefly describe the *Unsupervised $k$-Windows algorithm* (UKW).

As previously mentioned, UKW assumes that the dataset contains numeric attributes solely. Thus, if $A \subset \mathbb{R}^d$ denotes a $d$–dimensional dataset, the UKW algorithm utilizes $d$–dimensional ranges in $\mathbb{R}^d$ (windows) over $A$, to identify the underlying clusters. A $d$–dimensional window $Q$ is defined as the subset of $\mathbb{R}^d$: $Q = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_d, b_d]$ with $a_j \leqslant b_j$.

The first step of the algorithm is the initialization of a number of windows over the dataset. Subsequently, by identifying the points that are enclosed in each window, the algorithm employs three procedures to produce the clustering result:

(1) **Movement**: This procedure aims at iteratively positioning each window as close as possible to the center of a cluster. During this procedure each window is centered at the mean of the patterns that it includes.
(2) **Enlargement**: The enlargement process tries to augment each window so as to include as many patterns from the cluster over which it is positioned, as possible. Thus, the range of each window, for each coordinate separately

is enlarged, as long as, a significant number of additional points is enclosed by the window.

(3) **Merging**: During this operation, for each pair of overlapping windows, the number of patterns that lie in their intersection is computed. With respect to the proportion of this number to the total number of points contained in each window, the algorithm can decide whether:
   (a) Ignore one window if the proportion is very high.
   (b) Consider the windows to contain parts of the same cluster if the proportion is relatively high.
   (c) Consider the windows to capture different clusters, if the proportion is low.

By iteratively executing the movement and enlargement procedures, until they cease to alter any of the windows, and subsequently executing the merging procedure, the algorithm identifies the clusters and provides an approximation to their number. In Fig. 1 the three processes are illustrated.

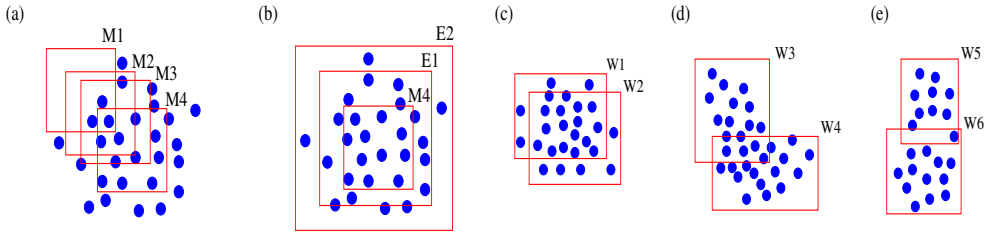


Fig. 1. (a) Sequential movements M2, M3, M4 of initial window M1. (b) Sequential enlargements E1, E2 of window M4. (c) W1 and W2 have a very high overlapment proportion thus W1 is discarded. (d) W3 and W4 have a significant overlapment and are considered to belong to the same cluster. (e) W5 and W6 have a small overlapment and capture two different clusters.

An example of the overall workings of the algorithm is presented in Fig. 2. In Fig. 2(a) six windows initialized over a dataset that contains three clusters are shown. In Fig. 2(b), after the merging operation has terminated, the algorithm correctly identifies the three clusters.

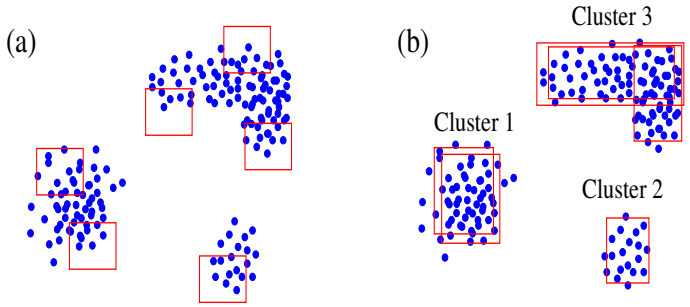


Fig. 2. An example of the application of the $k$-Windows algorithm.

The computationally demanding step of UKW is the determination of the

points that lie in a specific window. This is the well studied *orthogonal range search* problem [26]. Numerous Computational Geometry techniques have been proposed [26–29] to address this problem. All these techniques employ a preprocessing stage at which they construct a data structure that stores the patterns. This data structure allows them to answer range queries fast. In detail, for applications of very high dimensionality, data structures like the Multidimensional Binary Tree [26], and Bentley and Maurer [28] are appropriate. On the other hand, for cases with a large number of low dimensional points the approach of Alevizos [27] appears more attractive.

## 4 Generalized Unsupervised $k$-Windows Algorithm

In this Section we present the *Generalized Unsupervised k-Windows Algorithm* (GUKW) that is designed to extend UKW in metric spaces. The proposed generalization uses the algorithmic procedure of UKW but the data structures employed are appropriate to the domain of the metric space. Subsequently, if we denote with $(X, d)$ a metric space, where $X$ is a set of objects and $d : X \to \mathbb{R}$ is a metric distance function, then a range $Q$ of size $r$ is defined only around an object $c \in X$ (center of the range), and it is defined as $Q = \{y \in X : d(c, y) \leqslant r\}$. Note that the size of a range is determined by a single real value $r$.

Consequently, each of the three fundamental procedures of UKW, namely movement, enlargement, and merging, need to be modified to be applicable to metric spaces. In the following we analyze each procedure separately.

**Movement Procedure:** As previously mentioned, the movement procedure aims at re-centering each window to the mean of the underlying cluster. The assumption of a metric space imposes on the algorithm to select as mean one of the objects in $X$. Thus, in this case the window is re-centered around an objects $c \in Q$ that minimizes the equation:

$$D(c, Q) = \sum_{\substack{\forall x \in Q \\ x \neq c}} d(x, c). \tag{1}$$

This step affects the complexity of the algorithm. The computation of the minimizer of Eq. (1), through a brutal force algorithm imposes a quadratic time complexity, with respect to the number of points included in each window. Huang proposed in [21], a similarity metric that enables the computation of the minimizer (*mode* using Huang's terminology) in linear time. Due to its lower time complexity this approach is employed by the proposed algorithm.

In detail, following [21], we assume that the $X$ is composed of $m$ categorical attributes, and we employ the following metric:

$$d(x, y) = \sum_{i=1}^{m} \delta(x_i, x_j) \quad \forall x, y \in X,$$

where,

$$\delta((x_i, x_j) = \begin{cases} 0 \ (x_i = y_j) \\ 1 \ (x_i \neq y_j) \end{cases}.$$

Next, replace the centers of the windows with modes defined as:

**Definition 1** *A mode of a set $Q = \{Q_1, \ldots, Q_n\}$, is a vector $C = [c_1, \ldots, c_m]$ that minimizes $D(c, Q) = \sum_{i=1}^{n} d(Q_i, C)$.*

In the above definition, we do not restrict the mode of the set to be one of the objects in $Q$. The computation of the mode, is performed using the following theorem [21]:

**Theorem 2** *Let $n_{v_{i,j}}$, be the number of objects of a set $Q$, having the $i$th value $v_{i,j}$, for attribute $A_j$, and $f_r(A_j = v_{i,j}|Q) = n_{v_{i,j}}/|Q|$ represent the relative frequency of value $v_{i,j}$ in $Q$. The function $D(c, Q)$ is minimized iff $f_r(A_j = c_{i,j}|Q) \geqslant f_r(A_j = v_{i,j}|Q)$ for $c_j \neq v_{i,j}$ for all $j = 1, \ldots, m$.*

This theorem enables the computation, of the mode of a set in linear time, and can be easily incorporated in the $k$-Windows algorithm. In particular, a window's center at each successive movement is computed as the mode of the set of points included in it. The movement procedure terminates, when the distance between the mode of the included points and the center of the window is lower than a user defined threshold $\theta_v$.

**Enlargement Procedure:** The enlargement procedure has a straightforward generalization to the metric space case. An enlargement of a window with center $c$ and of size $r$ is possible only by enlarging $r$. In detail, $r$ is increased by a user defined proportion, $\theta_e$. Next, the increase in the number of points included in the window is calculated. If this increase exceeds $\theta_e \pm \delta$, then the enlargement is considered valid. Otherwise, the enlargement and movement steps are rejected and the center and size of the window are reverted to their prior to enlargement values. Note that before each enlargement is examined for validity the movement procedure is invoked.

**Merging Procedure:** The algorithm through this procedure is able to provide an approximation to the number of clusters in the dataset. This is achieved as in the numeric attribute case, through the examination of the proportion of common points for every pair of overlapping windows, with respect to the total number of points included in each window. The possible overlapping between two windows can be easily determined through the triangle inequality since we refer to metric spaces. Thus, two windows $W_i, W_j$ with centers $c_i, c_j$ and sizes $r_i, r_j$, overlap if $d(c_i, c_j) \leqslant r_i + r_j$. Next, as in the numeric attribute case, the points in the intersection are counted, and using two user defined thresholds, $\theta_s$ (similarity threshold) and $\theta_m$ (merging threshold), the merging decision is made. Specifically, if the proportion of the number of common points to the total number of points contained in each window is larger, than $\theta_s$ the window containing the fewer points is ignored. Otherwise, if it is larger than $\theta_m$ the windows are considered to contain parts of the same cluster. In the case that the proportion, is even smaller than $\theta_m$ the windows are considered to capture different clusters.

### *4.1 Computational Complexity Issues*

The computational complexity of the GUKW algorithm, is determined by the cost of discovering the points that reside in a window. The computational geometry approaches for the numeric domain case are not directly applicable to metric spaces. This is because some essential geometric concepts, such as minimum bounding region and the area of region, are not valid for the metric spaces [30]. Various methods specifically designed for metric spaces have been proposed [31]. These methods can be categorized into two classes. In the first class an index data structure is directly constructed from the distances among the objects; while in the second class the objects are mapped in a vector space. Experimental results for these methods [30,32–35], indicate that range queries can be answered in sub-linear time with respect to the database size.

## 5 Presentation of Experiments

The evaluation of the results of a clustering algorithm, relies on the assumption that there is an inherent grouping structure in the database. The degree to which the clustering results verify this assumption and recover the underlying grouping structure measures their efficiency. In this work we employ the *external criterion* discussed in [36]. This criterion, also used in [21], measures the degree of correspondence between the resulting clusters and the classes assigned a priori to each object. Therefore, the clustering accuracy, $r$, can be

defined as:

$$r = \frac{1}{n}\sum_{i=1}^{c} a_i,$$

where, $c$ stands for the number of clusters, $a_i$ denotes the number of objects of the class with the largest representation in cluster $i$; and $n$ is the total number of objects in the database.

The datasets used in this study include the *Soybean Disease Data Set* and the *1984 United States Congressional Voting Records Database*. Both datasets originate from the UCI Machine Learning Repository [37]. These datasets have been used for the evaluation of a number of similar algorithms [21,38,39].

### *5.1 The Soybean Disease Data Set*

The soybean data set consists of 47 records of 35 attributes. Each record is labeled as one of four diseases: Diaporthe Stem Canker, Charcoal Rot, Rhizoctonia Root Rot, and Phytophthora Rot. Each disease is represented by 10 records, with the exception of Phytophthora Rot that is represented by 17. From the 35 attributes only 21 have more than one different value, and thus only these are considered by the clustering procedure.

The algorithm was applied on this dataset with an initial window size equal to 4. The values of the other parameters were set to $1.0, 0.1, 0.1, 0.2, 0.8$ for $\theta_v$, $\theta_e$, $\theta_c$, $\theta_m$, and $\theta_s$, respectively.

The determination of the centers of the initial ranges can be performed through various approaches. The simplest approach involves the random selection of points from the database as centers. It is also possible to select the nodes of the index data structures, used for range searching, as initial points. In our implementation we used as an index structure the *Vantage Point Tree* [40]. Specifically, we selected a breadth–first selection of the nodes of the tree, until all windows were allocated. Moreover, we conducted 100 executions of the GUKW algorithm using randomly selected points as centers to compare the two initialization schemes. The misclassification matrices presented in Table 1 report the results of the GUKW algorithm using the vantage-point tree initialization with four initial windows. The labels PR, DSC, CR, and RRR, correspond to the four diseases Phytophthora Rot, Diaporthe Stem Canker, Charcoal Rot, and Rhizoctonia Root Rot, respectively. This initialization scheme resulted in an accuracy of 100%, which was unaltered by increasing the number of initial windows. In the boxplots exhibited in Fig. 3, we summarize the results of the partitioning accuracy of the random initialization method. Each boxplot

depicts the obtained values for the classification accuracy, in the 100 experiments. The box has lines at the lower quartile, median, and upper quartile values. The lines extending from each end of the box (whiskers) exhibit the range covered by the remaining data. The outliers, i.e. the values that lie beyond the ends of the whiskers, are represented with crosses. Notches represent a robust estimate of the uncertainty about the median.

As exhibited in Fig. 3, using random initialization the GUKW algorithm was able to achieve the 100% classification accuracy achieved by the vantage point tree method. As the number of initial windows increases the median classification accuracy also increases, and for 16 and 32 initial windows the probability of achieving the perfect 100% accuracy becomes very high.

| | Disease | | | |
|---|---|---|---|---|
| | PR | DSC | CR | RRR |
| Cluster 1 | 17 | 0 | 0 | 0 |
| Cluster 2 | 0 | 10 | 0 | 0 |
| Cluster 3 | 0 | 0 | 10 | 0 |
| Cluster 4 | 0 | 0 | 0 | 10 |
| | Classification Accuracy: 100% | | | |

Table 1
Misclassification matrix for the soybean dataset using 4 initial windows through the vantage point tree initialization method.

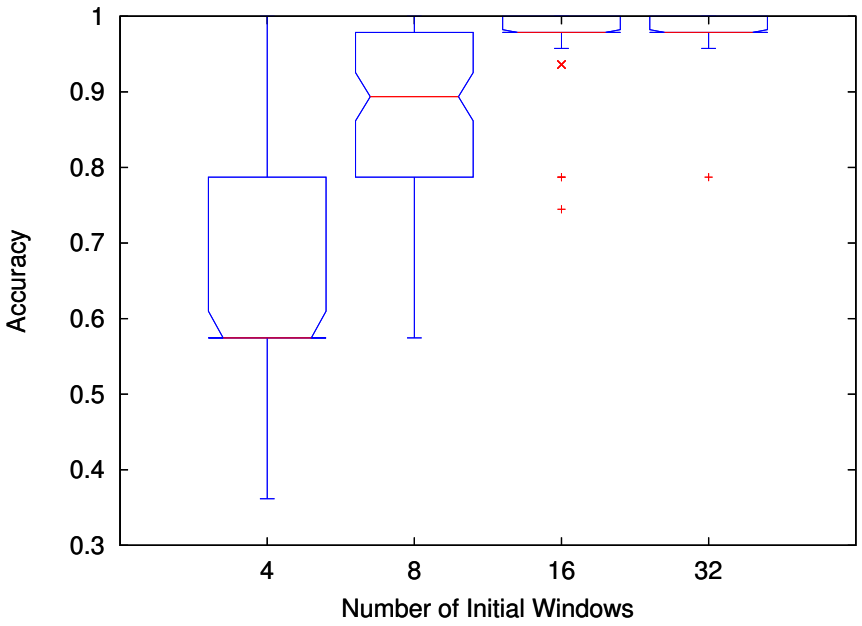


Fig. 3. Classification accuracy of GUKW using the random initialization with 4,8,16 and 32 initial windows.

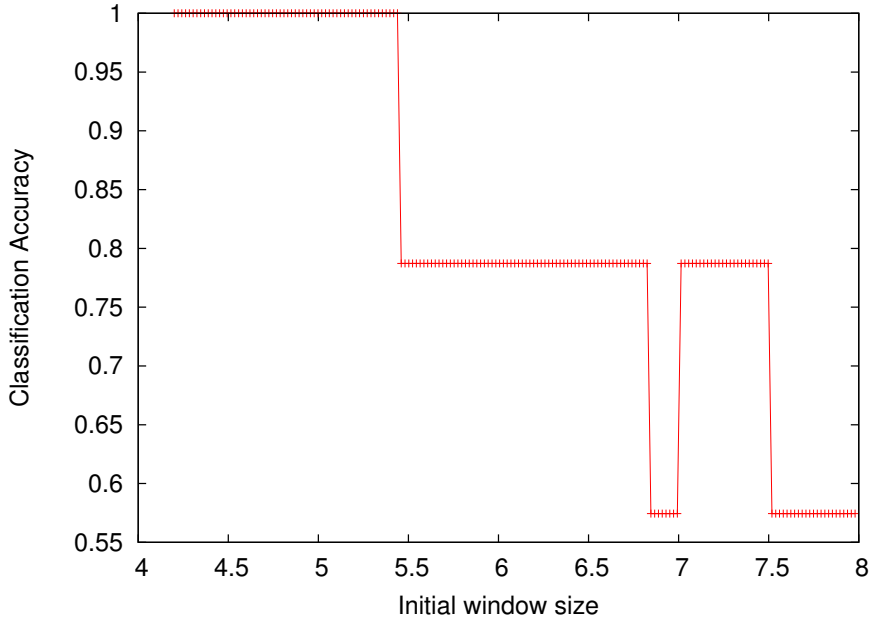


Fig. 4. Classification accuracy of GUKW for varying initial window size.

The results of the GUKW were stable with respect to the parameters, with the only exception being the size of the initial window, the choice of which bears a significant impact. To this end, in Fig. 4 we exhibit the classification accuracy obtained for all the values in the range $[4, 8]$ with 0.2 step.

For comparative purposes, we conducted experiments for the GDBSCAN, and the Principal Direction Divisive Partitioning (PDDP) [41] clustering algorithms. PDDP algorithm, is based on PCA, and hence, can address effectively problems involving high dimensional and sparse data. PDDP, as well as, PDDP($l$) [42] a recent generalization of PDDP, do not provide a direct estimation for the number of clusters. Proposed methods that provide such estimations with these algorithms are based on scattering of the data around their centroids. Nonetheless, these methods tend to overestimate the true number of clusters resulting in rigid clustering. In Table 2, we present the misclassification matrices for these two algorithms. The parameters for the GDBSCAN were set to $Epts = 2.0, Minpts = 4$, (refer to [24] for a thorough discussion of the algorithm) since these provided the best results after extensive experimentation. The PDDP algorithm was forced to return 4 clusters, but even for larger numbers of clusters the classification accuracy did not improve. GDBSCAN results in 5 clusters with classification accuracy of 100% but for 91% of the objects in the database since it regards 4 objects as outliers.

GUKW on the soybean dataset managed to provide high partitioning accuracy as the correspondence of each cluster to the class label suggests. Moreover, the initialization using the vantage point tree seems to provide a robust way of initializing cluster centers. The algorithm can yield an equivalent accuracy

| | GDBSCAN | | | | PDDP | | | |
|---|---|---|---|---|---|---|---|---|
| | Disease | | | | Disease | | | |
| | PR | DSC | CR | RRR | PR | DSC | CR | RRR |
| Cluster 1 | 0 | 10 | 0 | 0 | 9 | 0 | 0 | 5 |
| Cluster 2 | 0 | 0 | 10 | 0 | 8 | 0 | 0 | 5 |
| Cluster 3 | 0 | 0 | 8 | 0 | 0 | 0 | 10 | 0 |
| Cluster 4 | 11 | 0 | 0 | 0 | 0 | 10 | 0 | 0 |
| Cluster 5 | 4 | 0 | 0 | 0 | | | | |
| Outliers | 2 | 0 | 0 | 2 | | | | |
| | Classification Accuracy: 100% | | | | Classification Accuracy: 78.72% | | | |

Table 2
Misclassification matrix for the soybean dataset for the GDBSCAN and PDDP clustering algorithms.

through a random initialization of the windows' centers, but in this case a larger number of windows is required.

### *5.2 The 1984 United States Congressional Voting Records Database*

This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise made a position known (these three simplified to an unknown disposition). Thus, the data are described by 16 attributes. The total number of objects is 435, 267 of which are labeled as *democrats*, while the remaining 168 as *republicans*.

Similarly with the soybean dataset 100 experiments were performed with the random initialization step. The results are summarized in the boxplot exhibited in Fig. 5. As shown, for more than 16 initial windows the classification accuracy stabilizes around 90%. In Table 3, the results of GUKW are exhibited utilizing the vantage point tree and 32 initial windows. GUKW manages to discover two clusters that contain mostly objects of one of the two parties. Three more clusters are discovered that contain mostly Democrat labeled objects. Furthermore, one of the identified clusters contains only 3 objects, and can be considered as an outlier. The effect of the initial window size, on the classification accuracy, is demonstrated in Fig 6.

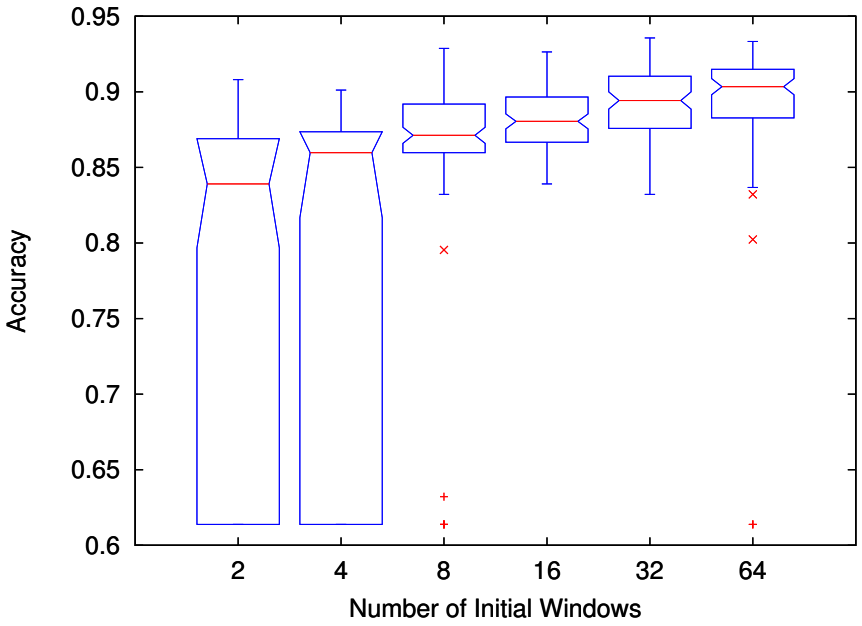


Fig. 5. Classification accuracy of GUKW using the random initialization with 2,4,8,16,32 and 64 initial windows.

Finally, to provide a comparative evaluation of the performance of the algorithm on this dataset we present the results from the application of the GDBSCAN and the PDDP algorithms in Table 4. The parameters for GDBSCAN were set to $Epts = 3.0, Minpts = 20$, as these values resulted in the highest accuracy. After experimenting with different values for the number of clusters, PDDP provided the best results for 5 clusters. The results of both GDBSCAN and PDDP indicate that the structure identified by the GUKW algorithm reflects accurately the nature of the data. For each party there exists one cluster with the highest concentration of objects that belong to this

| | Voter Party | |
|---|---|---|
| | Democrat | Republican |
| Cluster 1 | 28 | 6 |
| Cluster 2 | 26 | 7 |
| Cluster 3 | 8 | 1 |
| Cluster 4 | 21 | 148 |
| Cluster 5 | 1 | 2 |
| Cluster 6 | 183 | 4 |
| | Classification Accuracy: 90.84% | |

Table 3
Misclassification matrix using 32 initial windows through the vantage point tree initialization method.

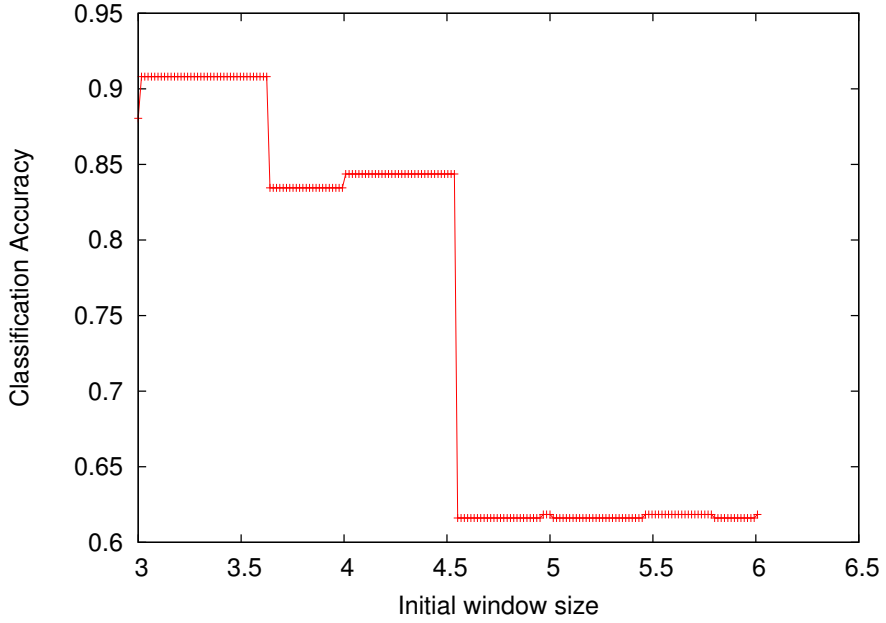


Fig. 6. Classification accuracy of GUKW for varying initial window size.

party. A number of clusters that mostly contain democrat labeled objects, (considered as outliers by the GDBSCAN algorithm) are spread in the data.

| | GDBSCAN | | PDDP | |
|---|---|---|---|---|
| | Voter Party | | Voter Party | |
| | Democrat | Republican | Democrat | Republican |
| Cluster 1 | 193 | 6 | 61 | 2 |
| Cluster 2 | 27 | 144 | 34 | 51 |
| Cluster 3 | | | 18 | 103 |
| Cluster 4 | | | 16 | 7 |
| Cluster 5 | | | 138 | 5 |
| Cluster 6 | | | | |
| Outliers | 47 | 18 | | |
| | Classification Accuracy: 90.11% | | Classification Accuracy: 84.83% | |

Table 4
Misclassification matrix for the voting records database for the GDBSCAN and PDDP clustering algorithms.

## 6 Concluding Remarks and Discussion

As database technologies advance, an increasing number of complex applications arise. A common characteristic is the description of each object with a mixture of numeric and discrete unordered values. Typical clustering algorithms assume that the data are composed of numerical data. Thus, the development of algorithms that can handle efficiently this complexity of the data is needed.

In this paper we presented a generalization of an unsupervised clustering algorithm for data in metric spaces. A critical issue in cluster analysis is the determination of the number of clusters in a database. The $k$-Windows algorithm has the capability to provide an approximation for this number. The proposed generalization is designed so as to retain this feature. To establish a fast running time for the algorithm we utilized techniques proposed in [21,31]. The experimental results on two real life datasets demonstrated that the algorithm is able to provide a high portioning accuracy. Moreover, by employing the index structure, constructed for fast query processing, as a guidance for the initialization the algorithm yielded a robust portioning performance.

## References


[1] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, Advances in Knowledge Discovery and Data Mining, MIT Press, 1996.

[2] M.S. Aldenderfer, R.K. Blashfield, Cluster Analysis, Vol. 44 of Quantitative Applications in the Social Sciences, SAGE Publications, London, 1984.

[3] V. Ramasubramanian, K. Paliwal, Fast $k$-dimensional tree algorithms for nearest neighbor search with application to vector quantization encoding, IEEE Transactions on Signal Processing 40 (3) (1992) 518–531.

[4] R.W. Becker, G.V. Lago, A global optimization algorithm, in: Proceedings of the 8th Allerton Conference on Circuits and Systems Theory, 1970, pp. 3–12.

[5] A. Törn, A. Žilinskas, Global Optimization, Springer-Verlag, Berlin, 1989.

[6] M. Rigou, S. Sirmakessis, A. Tsakalidis, A computational geometry approach to web personalization, in: IEEE International Conference on E-Commerce Technology (CEC'04), San Diego, California, 2004, pp. 377–380.

[7] C. Tryon, Cluster Analysis, Ann Arbor, MI: Edward Brothers, 1939.

[8] C. Linnaeus, Clavis Classium in Systemate Phytologorum in Bibliotheca Botanica, Amsterdam, The Netherlands: Biblioteca Botanica, 1736.

[9] D. K. Tasoulis, M. N. Vrahatis, Unsupervised Clustering on Dynamic Databases, Pattern Recognition Letters, (2005), in press.

[10] D.K. Tasoulis, M.N. Vrahatis, Unsupervised Clustering Using Fractal Dimension, International Journal of Biffurcation and Chaos, (2005), in press.

[11] M.N. Vrahatis, B. Boutsinas, P. Alevizos, G. Pavlides, The new $k$-Windows algorithm for improving the $k$-means clustering algorithm, Journal of Complexity 18 (2002) 375–391.

[12] D.K. Tasoulis, V.P. Plagianakos, M.N. Vrahatis, Unsupervised cluster analysis in bioinformatics, in: Fourth European Symposium on "Biomedical Engineering", 2004.

[13] D.K. Tasoulis, V.P. Plagianakos, M.N. Vrahatis, Unsupervised clustering of bioinformatics data, in: European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems, Eunite, 2004, pp. 47–53.

[14] G.D. Magoulas, V.P. Plagianakos, D.K. Tasoulis, M.N. Vrahatis, Tumor detection in colonoscopy using the unsupervised $k$-windows clustering algorithm and neural networks, in: Fourth European Symposium on "Biomedical Engineering", 2004.

[15] D.K. Tasoulis, L. Vladutu, V.P. Plagianakos, A. Bezerianos, M.N. Vrahatis, On-line neural network training for automatic ischemia episode detection, in: Leszek Rutkowski, Jörg H. Siekmann, Ryszard Tadeusiewicz, Lotfi A. Zadeh (Eds.), Lecture Notes in Computer Science, Vol. 2070, Springer-Verlag, 2003, pp. 1062–1068.

[16] N.G. Pavlidis, D.K. Tasoulis, V.P. Plagianakos, M.N. Vrahatis, Computational Intelligence Methods for Financial Time Series Modeling, International Journal of Biffurcation and Chaos, (2005), accepted for publication.

[17] N.G. Pavlidis, D.K. Tasoulis, M.N. Vrahatis, Financial forecasting through unsupervised clustering and evolutionary trained neural networks, in: Congress on Evolutionary Computation, Canberra Australia, 2003, pp. 2314–2321.

[18] J.A. Hartigan, M.A. Wong, A $k$-means clustering algorithm, Applied Statistics 28 (1979) 100–108.

[19] K. C. Gowda, E. Diday, Symbolic clustering using a new dissimilarity measure, Pattern Recogn. 24 (6) (1991) 567–578.

[20] J. C. Gower, A general coefficient of similarity and some of its properties, Biometrics 27 (1971) 857–871.

[21] Z. Huang, Extensions to the $k$-means algorithm for clustering large data sets with categorical values, Data Min. Knowl. Discov. 2 (3) (1998) 283–304.

[22] H. Ralambondrainy, A conceptual version of the k-means algorithm, Pattern Recogn. Lett. 16 (11) (1995) 1147–1157.

[23] R.T. Ng, J. Han, Clarans: A method for clustering objects for spatial data mining, IEEE Transactions on Knowledge and Data Engineering 14 (5) (2002) 1003–1016.

[24] J. Sander, M. Ester, H.-P. Kriegel, X. Xu, Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications, Data Mining and Knowledge Discovery 2 (2) (1998) 169–194.

[25] M. Ankerst, M. M. Breunig, H.-P. Kriegel, J. Sander, Optics: ordering points to identify the clustering structure, in: 1999 ACM SIGMOD international conference on Management of data, ACM Press, 1999, pp. 49–60.

[26] F. Preparata, M. Shamos, Computational Geometry, Springer Verlag, New York, Berlin, 1985.

[27] P. Alevizos, An algorithm for orthogonal range search in $d \geqslant 3$ dimensions, in: Proceedings of the 14th European Workshop on Computational Geometry, Barcelona, 1998.

[28] J.L. Bentley, H.A. Maurer, Efficient worst-case data structures for range searching, Acta Informatica 13 (1980) 155–168.

[29] B. Chazelle, Filtering search: A new approach to query-answering, SIAM J. Comput 15 (3) (1986) 703–724.

[30] G. Qian, Q. Zhu, Q. Xue, S. Pramanik, The nd-tree: A dynamic indexing technique for multidimensional non-ordered discrete data spaces, in: 29th International Conference on Very Large Data Bases, Berlin, Germany, 2003, pp. 620–631.

[31] G. R. Hjaltason, H. Samet, Index-driven similarity search in metric spaces, ACM Trans. Database Syst. 28 (4) (2003) 517–580.

[32] T. Bozkaya, M. Ozsoyoglu, Indexing large metric spaces for similarity search queries, ACM Transactions on Database Systems 24 (3) (1999) 361–404.

[33] B. Cui, B. C. Ooi, J. Su, K.-L. Tan, Indexing high-dimensional data for efficient in-memory similarity search, IEEE Transactions on Knowledge and Data Engineering 17 (3) (2005) 333–353.

[34] V. Dohnal, C. Gennaro, P. Savino, P. Zezula, D-index: Distance searching index for metric data sets, Multimedia Tools Appl. 21 (1) (2003) 9–33.

[35] G. Navarro, Searching in metric spaces by spatial approximation, The VLDB Journal 11 (1) (2002) 28–46.

[36] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice Hall, 1988.

[37] C.L. Blake, C.J. Merz, UCI repository of machine learning databases (1998).

[38] D.H. Fisher, Knowledge acquisition via incremental conceptual clustering, Mach. Learn. 2 (2) (1987) 139–172.

[39] R.S. Michalski, R.E. Stepp, Automated construction of classifications: Conceptual clustering versus numerical taxonomy, IEEE Transactions on Pattern Analysis and Machine Intelligence 5 (4) (1983) 396 – 409.

[40] N.P. Yianilos, Data structures and algorithms for nearest neighbor search in general metric spaces, in: ACM-SIAM Symposium on Discrete Algorithms, ACM Press, 1993, pp. 311–321.

[41] D. Boley, Principal direction divisive partitioning, Data Mining and Knowledge Discovery 2 (4) (1998) 325–344.

[42] D. Zeimpekis, E. Gallopoulos, PDDP($l$): Towards a Flexing Principal Direction Divisive Partitioning Clustering Algorithms, in: D. Boley, I. Dhillon, J. Ghosh, J. Kogan (Eds.), Proc. IEEE ICDM '03 Workshop on Clustering Large Data Sets, Melbourne, Florida, 2003, pp. 26–35.