An Attempt to Dynamically Process Multi-dimensional Data

Yong Shi, Brian Graham and Marcus Judd

Department of Computer Science and Information Systems
Kennesaw State University, 1000 Chastain Road, Kennesaw, GA 30144

Abstract

Cluster and outlier detection has always been one of data mining research interests. Numerous approaches have been designed to find clusters and detect outliers in various types of data sets. In this paper, we present our research on analyzing data sets with constant changes. We design approaches to keep track of status of clusters, the movement of data points, and the updated group of outliers. Different from the traditional approaches which are focused on two-dimensional or low-dimensional data spaces, we aim to analyze data sets in multi-dimensional data spaces. We also propose to adjust the clusters and outliers simultaneously, since they are two concepts that are closely related.

Keywords: *Dynamic data, multi-dimensional data*

1. Introduction

Everyday a large amount of real data sets are generated in many disciplines. Data mining approaches are designed to analyze those data sets. Cluster and outlier detection has always been one of the focuses of data mining research. Cluster analysis specializes in techniques for grouping similar objects into a cluster in which objects inside a cluster exhibit certain degree of similarities, and separates dissimilar objects into different clusters. It is a method of unsupervised learning, and a common technique for statistical data analysis used in many fields, including machine learning, data mining, pattern recognition, image analysis and bioinformatics. Existing clustering algorithms can be broadly classified into four types: partitioning [10–12], hierarchical [21, 7, 8], grid-based [18, 15, 3], and density-based [5, 9, 4] algorithms.

Partitioning algorithms construct a partition of a database of n objects into a set of K clusters, where K is an input parameter. In general, partitioning algorithms start with an initial partition and then use an iterative control strategy to optimize the quality of the clustering results by moving objects from one group to another. Hierarchical algorithms create a hierarchical decomposition of the given data set of data objects. The hierarchical decomposition is represented by a tree structure, called dendrogram. Grid-based algorithms quantize the space into a finite number of grids and perform all operations on this quantized space. These approaches have the advantage of fast processing time independent of the data set size and are dependent only on the number of segments in each dimension in the quantized space. Density-based approaches are designed to discover clusters of arbitrary shapes. These approaches hold that, for each point within a cluster, the neighborhood of a given radius must exceed a defined threshold. Density-based approaches can also filter out outliers.

Each of the existing clustering algorithms has both advantages and disadvantages. The most common problem is rapid degeneration of performance with increasing dimensions [9], particularly with approaches originally designed for low-dimensional data. To solve the high-dimensional clustering problem, dimension reduction methods [3, 2, 14] have been proposed which assume that clusters are located in a low-dimensional subspace.

An outlier is a data point that does not follow the main characteristics of the input data. Outlier detection is concerned with discovering the exceptional behaviors of certain objects. It is an important branch in the field of data mining with numerous applications, including credit card fraud detection,

discovery of criminal activities, discovery of computer intrusion, etc. In many applications outlier detection is at least as significant as cluster detection. There are numerous studies on outlier detection [19, 17, 20, 13].

In this paper, we analyze data sets with constant changes. We design approaches to keep track of status of clusters, the movement of data points, and the updated group of outliers. Different from the traditional approaches which are focused on two-dimensional or low-dimensional data spaces, we aim to analyze data sets in multi-dimensional data spaces. We also propose to adjust the clusters and outliers simultaneously, since they are two concepts that are closely related.

2. Related Work

Numerous approaches have been designed to analyze data sets with constant changes. For example, Abrantes etc. [1] proposed a data clustering method that extends well-known static clustering algorithms, applying a motion model to track clusters that deform and translate. The method uses centroids, or points of reference within the data cluster that are drawn towards the center of clusters. The clusters are defined by their relevance to the centroids. In every instance of tracking the previous centroids are used to calculate translations and deformations of the cluster, and establish new centroids, based on previous calculations. They also demonstrated examples of this process in the context of object tracking using pixels and three dimensional linear calculations.

Garcia etc. [6] proposed a method for clustering data with a dissimilarity measure and a dynamic procedure of splitting, giving examples of the method by using plot graphs of two-dimensional data sets. The dissimilarity measure uses the optimum path between each successive datum. The optimum path is chosen by finding the shortest distance between two successive vertices. This measure allows the clusters to take a unique shape, rather than clustered into quadrants. The optimal partitioning can be performed using the previous optimum path as a comparison, so clusters that are dense will be less likely to assume outliers or data belonging to another cluster. The authors also described a method to scale and smooth the derivatives.

Irpino etc. [23] proposed algorithms to deal with improving the adaptive distances of clustering data in order to overcome the variability of clusters and also the variables within the clusters, based on the pre-established Wasserstein distance of clusters.

Gabrielov etc. [24] presented an inverse cascade model for similar growth clusters using branches to describe the coalescence between two or more clusters, with variables within clusters assigned certain ranks.

Our approach is different from the previous work in that, instead of solely focusing on clustering analysis, we keep track of the change of clusters and outliers, and always keep them in the most updated status. The characteristic of certain data points in clusters and certain outliers are also changed dynamically. Furthermore, we design our approach in multi-dimensional data spaces instead of two-dimensional or low-dimensional data spaces.

3. Analyzing Dynamic Data Sets

A lot of algorithms have been designed for cluster analysis and outlier detection. It is difficult to detect clusters and outlier with a high accuracy for multi-dimensional noisy data sets, especially when the data sets change constantly. For example, Figure 1 shows a two-dimensional data set whose data points move dynamically over the time. The directions of the movement for certain data points are unpredictable.

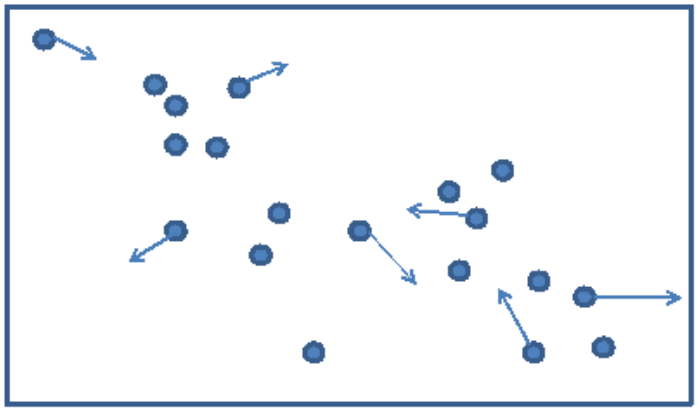

Figure 1. An Example of Dynamic Data Set

In this section we discuss how to design an approach to keep track of the status of clusters, the movement of data points, and the updated group of outliers. In order to describe our approaches, we shall introduce a few notations and definitions. Let *n* denote the total number of data points and *d* be the dimensionality of the data space. Let D_k be the *k*th dimension, where k = 1, 2, ..., d. Let the input *d*-dimensional data set be

$$\boldsymbol{DS} = \{X_1, X_2, \ldots, X_n\},$$

which is normalized to be within the hypercube $[0, 1]^d \subset R^d$. Each data point *Xi* is a *d*-dimensional vector:

$$X_i = [x_{i1}, x_{i2}, \ldots, x_{id}]. \quad (1)$$

Our approach is designed to keep track of the variation of clusters and outliers for a give data set with multiple dimensions. For a given step, let the current number of clusters be *kc* and the current number of outliers be *ko*; let the set of clusters be C = {C_1 , C_2 , ..., C_{kc} }, and the set of outliers be O = {O_1, O_2 , ..., O_{ko}}.

For each cluster $Ci \in C$, i=1,2,...,*kc*, we define its size. For data sets in a two-dimensional space, a traditional way is to use the radius to represent how large a cluster is:

$$radius\,(C_i) = \max_{X_p \in C_i}(d(X_p, m_{c_i})), \quad (2)$$

where m_{ci} is the centroid of Cluster *Ci* , *Xp* is any data point in Cluster *Ci*, and $d(Xp, m_{ci})$ is the distance between *Xp* and m_{ci} under certain distance metric, normally Euclidean distance for two- dimensional data space.

However, as the dimensionality of the data space goes higher, the radius of a cluster will increase dramatically. This is because as shown in equation 2, the distance between a data point in a cluster and its centroid is calculated by $d(Xp, m_{ci})$ using Euclidean distance, and it is well known that Euclidean distance increases very fast when the dimensionality goes higher. This is called the "curse of dimensionality".

We can also analyze the case in another way. In a two-dimensional data space, a cluster is represented by a circle. The volume of the circle for a cluster can be calculated as πr^2, where r is the radius of the cluster. In a multi-dimensional data space, a cluster will be represented by

a hyper-sphere *S*. There may be many empty regions which contain no data, and the bounding hyper-spheres of two different clusters may overlap.

The volume v of the hyper-sphere *S* in a d-dimensional data space is calculated as

$$v = \frac{2\pi^{d/2} r^d}{d\Gamma(\frac{d}{2})}, \tag{3}$$

The gamma function $\Gamma(x)$ is defined as

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt, \tag{4}$$

where $\Gamma(x+1) = x\Gamma(x)$ and $\Gamma(1) = 1$.

From equation 3 and equation 4 we can see that the volume v of the hyper-sphere *S* for a cluster increases dramatically as dimensionality goes higher.

Here we apply a different approach to define the size of a cluster. Let D_k be the *k*th dimension, where k = 1, 2, ..., d, the lower bound of the value range on D_k be the smallest value of the data points on D_k, and the upper bound of the value range on D_k be the largest value of the data points on D_k . For a given data set DS in a *d*-dimensional data space, we represent the size of a cluster C in DS as a group of intervals:

$$Size(C) = \{[l_1, h_1], [l_2, h_2], \ldots, [l_d, h_d]\}, \tag{5}$$

where l_k and h_k are the lower bound and upper bound of the value range on D_k , k = 1, 2, ..., d. The reason we define the size of a cluster C in this way is that, when the dimensionality goes higher, the size of C will not increase dramatically like what the Euclidean distance causes. Instead, there will be just more pairs of lower bound and upper bound added in the size of the C.

In our approach, we closely keep track of the change of clusters and outliers based on the movement of the data points in a data set DS. Based on how fast the data points in DS change their positions and availabilities, a time interval *t* is assigned to DS.

At each time interval *t*:

1) For each cluster $C_i \in C$, we check the change of position for each data point $X_p \in C_i$, and count the number n_i of data points in C_i whose new value(s) on a certain dimension or certain dimensions are out of the value intervals defined in $\{[l_1, h_1], [l_2, h_2], \ldots, [l_d, h_d]\}$. If a data point no longer exists in DS, i.e., it is deleted from DS, it will also be counted into n_i.

   If n_i exceeds a certain threshold, we will modify the intervals in $\{[l_1, h_1], [l_2, h_2], \ldots, [l_d, h_d]\}$ so it will still contain those data points, because in this case the majority of data points in C_i are moving out of the range of C_i, thus the size and shape of C_i need to be adjusted to still form a valid cluster. If n_i does not exceeds the threshold, which means those data points are the minority in C_i, they should be removed from C_i. For each of those data points, we will check to see if it resides in

the new range of other clusters in *C*. If it does, we will assign it to the new cluster, otherwise, we will assign it as a new outlier.

2) For each outlier O_j in *O*, we will check to see if the new position of O_j resides in the new range of a cluster C_q in *C*. If it does, we will assign O_j as a new data point in C_q, otherwise, O_j remains as an outlier.

The dynamic cluster-outlier adjustment algorithm is described in Figure 2.

Algorithm: Dynamic data set process
Begin
a) Generate the groups of clusters and outliers from the initial data set $\mathbf{DS} = \{X_1, X_2, ..., X_n\}$. Various algorithms can be adopted to perform the initial clustering and outlier detection step, such as [16], etc.
b) Define a time interval t based on the frequency of change for positions and availabilities of data points in DS.
c) Monitor and record the change of data points' positions and availabilities dynamically.
d) At each interval t, perform step 1) and 2) mentioned in the last subsection.
e) Keep performing the algorithm until the data set no longer changes or the user interrupts the process.
f) Output the currently updated data set DS, the current set of clusters $\mathcal{C} = \{C_1, C_2, ..., C_{k_c}\}$, and the current set of outliers $\mathcal{O} = \{O_1, O_2, ..., O_{k_o}\}$.
End.

Figure 2. Algorithm: Dynamic Data Set Process

3.1 Time and Space Analysis

Suppose the size of the data set is n and the dimensionality is *d*. Throughout the process, in each time interval, we need to keep track of the change of values of all points, which collectively occupies *O*(*dn*) space.

In each time interval, and for each cluster, we need to calculate the value n_i which is the number of data points whose values are out of the value intervals of its cluster. The time required for this process is *O*(*dn*). Suppose there are T_n intervals before the algorithm is terminated. The processing time is $O(T_n dn)$.

4. Experiments

Various experiments were performed to evaluate and demonstrate the effectiveness and efficiency of the proposed approach. Our experiments were run on Intel(R) Pentium(R) 4 with CPU of 3.39GHz and Ram of 0.99 GB.

A synthetic data generator was generated to test the scalability of our algorithm over data size, dimensionality and time intervals. It produces data sets with normalized distributions. The sizes of the data sets vary from 2,500, 5,000, ... to 20,000, with the gap of 2,500 between each two adjacent data set sizes, and the dimensions of the data sets vary from 5, 10, ... to 40, with the gap of 5 between each two adjacent numbers of dimensions. To simulate the dynamic change of the data set, we applied the following strategies: 1) A time trigger was designed; 2) Every time the time trigger is randomly turned on: 2.1) A random subset RA of data points from the data set are selected to have their values changed, 2.2) A random subset RB of data points are removed from the data set, 2.3) A set RC of randomly generated data points are inserted into the data set. With these steps the data sets are constantly changing.

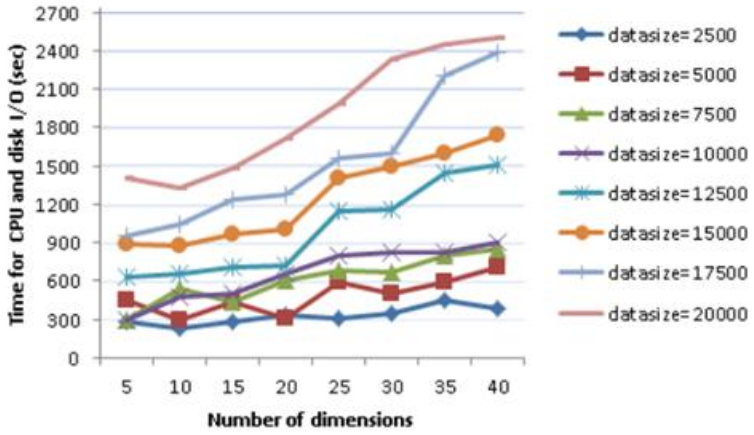

Figure 3. Running Time of the Algorithm on Data Sets with Increasing Dimensions

Figure 3 shows the running time of groups of data sets with dimensions increasing from 5 to 40.

Each group has a fixed data size (from 2,500, 5,000, ... to 20,000). And we set the time interval as 1 second.

Figure 4 shows the running time of groups of data sets with sizes increasing from 2,500 to 20,000. Each group has fixed number of dimensions (from 5, 10, ... to 40).

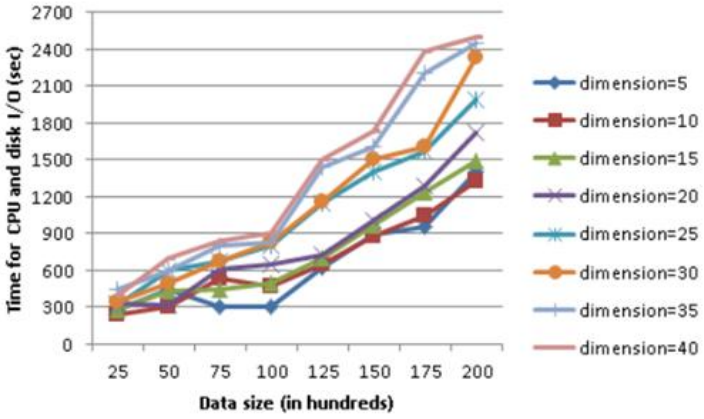

Figure 4. Running Time of the Algorithm on Data Sets with Increasing Sizes

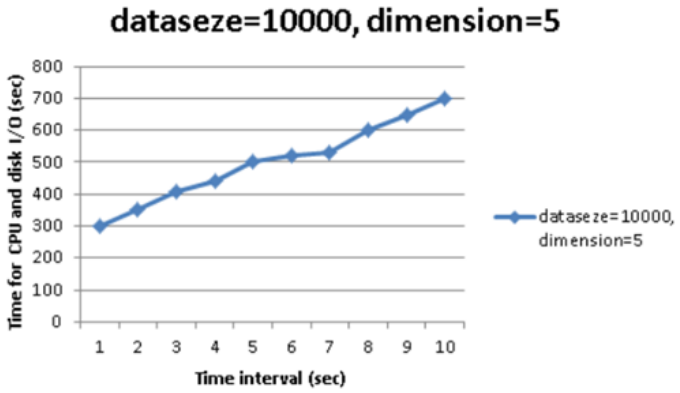

Figure 5. Running Time of the Algorithm on a Data Set with Increasing Time Intervals

And we set the time interval as 1 second. The two figures indicate that our algorithm is scalable over dimensionality and data size.

Figure 5 shows the running time of a data set with 10000 data points and 5 dimensions. The time interval changes from 1 to 10 seconds, with the gap of 1 second between each two adjacent time intervals. Figure 5 indicates that our algorithm is scalable over the time intervals.

We next evaluate the effectiveness of our proposed approach on read data sets obtained from UCI Machine Learning Repository [22]. We use 5 data sets with various format and content to demonstrate how approach works on different data.

The first one is the ionosphere data set, a radar data set collected by system in Goose Bay, Labrador. It contains 351 data points and each of data point has 34 dimensions. Two ground truth classes exist in the ionosphere data: *g* as *good*, and *b* as *bad*.

The second data set is Wine Recognition data set. It contains the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. It contains 178 instances. Each instance has 13 features which means the data set is defined in a 13 dimensional data space. Three clusters are difined with the sizes of 59, 71 and 48.

The third data set is the glass data set for different glass types. The glass data set contains 214 data points in a 9 dimensional data space. There are 7 classes in the glass data.

The fourth data set is Ecoli data set for Protein Localization Sites. There are 336 instances, each of which having 7 features. 8 clusters are contained in the data set.

The fifth data set is the well-known iris data set for various iris plant types. It contains 150 data points, and each of the data points has 4 dimensions. There are altogether 3 classes in the iris data: Irissetosa, Irisversicolor, and Irisvirginica.

We perform our algorithm on these 5 real data sets. The experimental accuracy on the ionosphere data set is 93.3%. The experimental accuracy on the ionosphere data set is 91.6%. The experimental accuracy on the ionosphere data set is 88.2%. The experimental accuracy on the ionosphere data set is 92.4%. The experimental accuracy on the ionosphere data set is 86.5%. From the experimental results we can see that our algorithm performs well on various data sets.

4. Conclusions

In this paper, we present a novel approach to analyzing the dynamic multi-dimensional data sets, which always keeps the clusters and outlier in the most updated status when the data points in the data sets change their positions and availabilities constantly. We will further conduct more experiments on synthetic and real data sets to test and demonstrate the efficiency and effectiveness of our approach.

References

[1] A. J. Abrantes and J. S. Marques, "A method for dynamic clustering of data", In BMVC, **(1998)**.
[2] C. C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu and J. Park, "Fast algorithms for projected clustering", In Proceedings of the ACM SIGMOD CONFERENCE on Management of Data, Philadelphia, PA, **(1999)**, pp. 61–72.
[3] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications", In Proceedings of the ACM SIGMOD Conference on Management of Data, Seattle, WA, **(1998)**, pp. 94–105.
[4] M. Ankerst, M. M. Breunig, H. -P. Kriegel and J. Sander, "OPTICS: Ordering Points To Identify the Clustering Structure", Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA, **(1999)**, pp. 49–60.
[5] M. Ester, K. H.-P. J. Sander and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise", In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, **(1996)**.
[6] J. A. Garc´ıa, J. Fdez-Valdivia, F. J. Cortijo and R. Molina, "A dynamic approach for clustering data", Signal Process., vol. 44, **(1995)** June, pp. 181–196.
[7] S. Guha, R. Rastogi and K. Shim, "Cure: An efficient clustering algorithm for large databases", In Proceedings of the ACM SIGMOD conference on Management of Data, Seattle, WA, **(1998)**, pp. 73–84.
[8] S. Guha, R. Rastogi and K. Shim, "Rock: A robust clustering algorithm for categorical attributes", In Proceedings of the IEEE Conference on Data Engineering, **(1999)**.
[9] A. Hinneburg and D. A.Keim, "An efficient approach to clustering in large multimedia databases with noise", In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, New York, **(1998)** August, pp. 58–65.
[10] J. MacQueen, "Some methods for classification and analysis of multivariate observations", Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, Statistics, **(1967)**.
[11] L. Kaufman and P. J. Rousseeuw, "Finding groups in data: an introduction to cluster analysis", **(1990)**.
[12] R. T. Ng and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining", In Proceedings of the 20th VLDB Conference, Santiago, Chile, **(1994)**, pp. 144–155.
[13] G. L. Peterson and B. T. McBride, "The importance of generalizability for anomaly detection", Knowl. Inf. Syst., vol. 14, no. 3, **(2008)**, pp. 377–392.
[14] T. Seidl and H. Kriegel, "Optimal multi-step k-nearest neighbor search", In Proceedings of the ACM SIGMOD conference on Management of Data, Seattle, WA, **(1998)**, pp. 154–164.
[15] G. Sheikholeslami, S. Chatterjee and A. Zhang, "Wavecluster: A multi-resolution clustering approach for very large spatial databases", In Proceedings of the 24th International Conference on Very Large Data Bases, **(1998)**.
[16] Y. Shi and A. Zhang, "Towards exploring interactive relationship between clusters and outliers in multi-dimensional data analysis", In International Conference on Data Engineering (ICDE), **(2005)**.

[17] Y. Tao, X. Xiao and S. Zhou, "Mining distance-based outliers from large databases in any metric space", In KDD, **(2006)**, pp. 394–403.
[18] W. Wang, J. Yang and R. Muntz, "STING: A Statistical Information Grid Approach to Spatial Data Mining", In Proceedings of the 23rd VLDB Conference, Athens, Greece, **(1997)**, pp. 186–195.
[19] M. Wu and C. Jermaine, "Outlier detection by sampling with accuracy guarantees", In KDD, **(2006)**, pp. 767–772.
[20] J. Yang, N. Zhong, Y. Yao and J. Wang, "Local peculiarity factor and its application in outlier detection", In KDD, **(2008)**, pp. 776–784.
[21] T. Zhang, R. Ramakrishnan and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases", In Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Canada, **(1996)**, pp. 103–114.
[22] S. D. Bay, The UCI KDD Archive [http://kdd.ics.uci.edu]. University of California, Irvine, Department of Information and Computer Science.
[23] A Irpino, R Verde and F Carvalho, "Dynamic Clustering of Histogram Data Based on Adaptive Squared Wasserstein Distances", In CoRR, abs/1110.1462, **(2011)**.
[24] A Gabrielov, W. I. Newman and D. L. Turcotte, "Exactly soluble hierarchical clustering model: Inverse cascades, self-similarity, and scaling", In Phys. Rev. E, vol. 60, Issue 5, **(1999)**, pp. 5293-5300.