# Tracking Clusters in Evolving Data Sets

*Daniel Barbará* *Ping Chen*
George Mason University *
Information and Software Engineering Department
Fairfax, VA 22303
{dbarbara,pchen}@gmu.edu

**Abstract**

As organizations accumulate data over time, the problem of tracking how patterns evolve becomes important. In this paper, we present an algorithm to track the evolution of cluster models in a stream of data. Our algorithm is based on the application of bounds derived using Chernoff's inequality and makes use of a clustering algorithm that was previously developed by us, namely Fractal Clustering, which uses self-similarity as the property to group points together. Experiments show that our tracking algorithm is efficient and effective in finding changes on the patterns.

## Introduction

Organizations today accumulate data at a astonishing rate. This fact brings new challenges for data mining. For instance, finding out when patterns change in the data opens the possibility of making better decisions and discovering new interesting facts. The challenge is to design algorithms that can track changes in an incremental way and without making growing demands on memory.

In this paper we present a technique to track changes in cluster models. Clustering is a widely used technique that helps uncovering structures in data that were previously not known. Our technique helps in discovering the points in the data stream in which the cluster structure is changing drastically from the current structure. Finding changes in clusters as new data is collected can prove fruitful in scenarios like the following:

- Tracking the evolution of the spread of illnesses. As new cases are reported, finding out how clusters evolve can prove crucial in identifying sources responsible for the spread of the illness.
- Tracking the evolution of workload in an e-commerce server (clustering has already been successfully used to characterize e-commerce workloads (Menascé *et al.* 1999)), which can help in dynamically fine tune the server to obtain better performance.
- Tracking meteorological data, such as temperatures registered throughout a region, by observing how clusters of spatial-meteorological points evolve in time.

Our technique is based on a novel clustering algorithm that we call *Fractal Clustering* (citation erased due to the anonymous reviewing process) which uses the notion of *self-similarity* to cluster points. The idea we exploit in this paper is to track the number of outliers that the next batch of points produce with respect to the current clusters, and with the help of analytical bounds decide if we are in the presence of data that does not follow the patterns (clusters) found so far. If that is the case, we proceed to re-cluster the points to find the new model. The technique has the virtue of being incremental by requiring only one pass over each data record, and by basing its decisions on the record being processed and on a **fixed** amount of information kept about the current clustering models. (Hence, the memory demands are kept bounded during the entire life of the process.) It is important to remark that other clustering algorithms previously published in the literature (e.g., K-means (Selim & Ismail 1984)) do not share the properties of begin incremental and having a concise representation of the clusters, and therefore are not suitable for the task we describe in this paper.

The paper is organized as follows. In Section 2 we provide the background of Fractal Clustering needed to understand the tracking technique. In Section 3, we derive the analytical bounds and present our tracking algorithm. In Section 4, we describe a simple, but representative experiment that illustrates how our technique works. Finally, Section 5 concludes the paper and delineates future avenues of research.

## Fractal Clustering

In this section we briefly describe our previously published Fractal Clustering algorithm (Barbará &

*This work has been supported by NSF grant IIS-9732113

Chen 2000), which we will use as the basis for tracking clusters.

Nature is filled with examples of phenomena that exhibit seemingly chaotic behavior, such as air turbulence, forest fires and the like. However, under this behavior it is almost always possible to find *self-similarity*, i.e. an invariance with respect to the scale used. The structures that exhibit self-similarity over every scale are known as *fractals* (Mandelbrot 1983). On the other hand, many data sets that are not fractal, exhibit self-similarity over a range of scales. It is important to remark that we do not require that the data sets we want to cluster are fractals, but rather that their clusters exhibit self-similarity over a range of scales.

Fractals have been used in numerous disciplines (for a good coverage of the topic of fractals and their applications see (Schroeder 1991)). In the database area, fractals have been successfully used to analyze R-trees (Faloutsos & Kamel 1997), Quadtrees (Faloutsos & Gaede 1996), model distributions of data (Faloutsos, Matias, & Silberschatz 1996) and selectivity estimation (Belussi & Faloutsos 1995).

Self-similarity can be measured using the *fractal dimension*. Loosely speaking, the fractal dimension measures the number of dimensions "filled" by the object represented by the data set. In truth, there exists an infinite family of fractal dimensions. By embedding the data set in an $n$-dimensional grid which cells have sides of size $r$, we can count the frequency with which data points fall into the $i$-th cell, $p_i$, and compute $D_q$, the generalized fractal dimension (Grassberger 1983; Grassberger & Procaccia 1983), as shown in Equation 1.

$$D_q = \begin{cases} \frac{\partial \log \sum_i p_i \log p_i}{\partial \log r} & \text{for } q = 1 \\ \frac{1}{q-1} \frac{\partial \log \sum_i p_i^q}{\partial \log r} & \text{otherwise} \end{cases} \quad (1)$$

Among the dimensions described by Equation 1, the *Hausdorff fractal dimension* ($q = 0$), the *Information Dimension* ($\lim_{q \to 1} D_q$), and the *Correlation dimension* ($q = 2$) are widely used. The Information and Correlation dimensions are particularly useful for data mining, since the numerator of $D_1$ is Shannon's entropy, and $D_2$ measures the probability that two points chosen at random will be within a certain distance of each other. Changes in the Information dimension mean changes in the entropy and therefore point to changes in trends. Equally, changes in the Correlation dimension mean changes in the distribution of points in the data set.

The traditional way to compute fractal dimensions is by means of the box-counting plot. For a set of $N$ points, each of $D$ dimensions, one divides the space in grid cells of size $r$ (hypercubes of dimension $D$). If $N(r)$ is the number of cells occupied by points in the data set, the plot of $N(r)$ versus $r$ in log-log scales is called the *box-counting plot*. The negative value of the slope of that plot corresponds to the Hausdorff fractal dimension $D_0$. Similar procedures are followed to compute other dimensions, as described in (Liebovitch & Toth 1989).

1.Given a batch $S$ of points brought to main memory:
2.For each point $p \in S$:
  3.For $i = 1, \cdots, k$:
    4. Let $C_i' = C_i \bigcup \{p\}$
    5. Compute $F_d(C_i')$
    6.Find $\hat{i} = min_i(|F_d(C_i' - F_d(C_i)|)$
    7.If $|F_d(C_{\hat{i}}') - F_d(C_{\hat{i}})| > \tau$
      8.Label $p$ as an outlier
    9.else
      10. Place $p$ in cluster $C_{\hat{i}}$

**Figure 1: The incremental step for FC.**

After we get the initial clusters, we can proceed to cluster the rest of the data set. Each cluster found by the initialization step is represented by a set of boxes (cells in a grid). Each box in the set records its population of points. Let $k$ be the number of clusters found in the initialization step, and $C = \{C_1, C_2, \ldots, C_k\}$ where $C_i$ is the set of boxes that represent cluster $i$. Let $F_d(C_i)$ be the fractal dimension of cluster $i$.

The incremental step brings a new set of points to main memory and proceeds to take each point and add it to each cluster, computing its new fractal dimension. The pseudo-code of this step is shown in Figure 1. Line 5 computes the fractal dimension for each modified cluster (adding the point to it). Line 6 finds the proper cluster to place the point (the one for which the change in fractal dimension is minimal). We call the value $|F_d(C_i' - F_d(C_i)|$ the *Fractal Impact* of the point being clustered over cluster $i$. The quantity $min_i|F_d(C_i' - F_d(C_i)|$ is the *Minimum Fractal Impact* (MFI) of the point. Line 7 is used to discriminate outliers. If the MFI of the point is bigger than a threshold $\tau$, then the point is simply rejected as an outlier (Line 8). Otherwise, it is included in that cluster. We choose to use the Hausdorff dimension, $D_0$, for the fractal dimension computation of Line 5 in the incremental step. We chose $D_0$ since it can be computed faster than the other dimensions and it proves robust enough for the task.

To compute the fractal dimension of the clusters every time a new point is added to them, we keep the cluster information using a series of grid repre-

sentations, or layers. In each layer, boxes (i.e., grids) have a size that is smaller than in the previous layer. The sizes of the boxes are computed in the following way. For the first layer (largest boxes), we divide the cardinality of each dimension in the data set by 2, for the next layer, we divide the cardinality of each dimension by 4 and so on. Accordingly, we get $2^D, 2^{2D}, \cdots, 2^{LD}$ $D$-dimensional boxes in each layer, where $D$ is the dimensionality of the data set, and $L$ the maximum layer we will store. Then, the information kept is not the actual location of points in the boxes, but rather, the number of points in each box. It is important to remark that the number of boxes in layer $L$ can grow considerably, specially for high-dimensionality data sets. However, we need only to save boxes for which there is any population of points, i.e., empty boxes are not needed. The number of populated boxes at that level is, in practical data sets, considerably smaller (that is precisely why clusters are formed, in the first place). Let us denote by $B$ the number of populated boxes in level $L$. Notice that, $B$ is likely to remain very stable throughout passes over the incremental step.

## Tracking clusters

As we get a new batch of points to be clustered we can ask ourselves if these points can be adequately clustered using the models we have so far. The key to answer this question is to count the number of outliers in this batch of points. A point is deemed an outlier in the test of Line 7, in Figure 1, when the MFI of the point exceeds a threshold $\tau$. We can use the Chernoff bound (Chernoff 1952) and the concept of adaptive sampling (Lipton *et al.* 1993; Lipton & Naughton 1995; Domingo, Gavaldá, & Watanabe 1998; 2000; Domingos & Hulten 2000), to find the minimum number of points that must be successfully clustered after the initialization algorithm in order to guarantee with a high probability that our clustering decisions are correct.

Let us define a random variable $X_i$, whose value is 1 if the $i$-th point to be clustered by FC has a MFI which is less than $\tau$, and 0 otherwise. Using Chernoff's inequality one can bound the expectation of the sum of the $X_i$'s, $X = \sum_i^n X_i$, which is another random variable whose expected value is $np$, where $p = Pr[X_i = 1$, and $n$ is the number of points clustered. The bound is shown in Equation 2, where $\epsilon$ is a small constant.

$$Pr[\ X/n\ >\ (1+\epsilon)p\ ]\ \leq\ exp(-pn\epsilon^2/3) \qquad (2)$$

Notice that we really do not know $p$, but rather have an estimated value of it, namely $\hat{p}$, given by the number of times that $X_i$ is 1 divided by $n$. (I.e., the number of times we can successfully cluster a point divided by the total number of times we try.) In order that the estimated value of $p$, $\hat{p}$ obeys Equation 3, which bounds the estimate close to the real value with an arbitrarily large probability (controlled by $\delta$), one needs to use a sample of $n$ points, with $n$ satisfying the inequality shown in Equation 4.

$$Pr[\ \|\hat{p}-p\|\ ]\ >\ 1-\delta \qquad (3)$$

$$n\ >\ \frac{3}{p\epsilon^2} ln(\frac{2}{\delta}) \qquad (4)$$

By using adaptive sampling, one can keep bringing points to cluster until obtaining at least a number of successful events (points whose minimum fractal impact is less than $\tau$) equal to $s$. It can be proven that in adaptive sampling (Watanabe 2000), one needs to have $s$ bound by the inequality shown in Equation 5, in order for Equation 3 to hold. Moreover, with probability greater than $1-\delta/2$, the sample size (number of points processed) $n$, would be bound by the inequality of Equation 6. (Notice that the bound of Equation 6 and that of Equation 4 are very close; The difference is that the bound of Equation 6 is achieved without *knowing* $p$ in advance.)

$$s\ >\ \frac{3(1+\epsilon)}{\epsilon^2} ln(\frac{2}{\delta}) \qquad (5)$$

$$n\ \leq\ \frac{3(1+\epsilon)}{(1-\epsilon)\epsilon^2 p} ln(\frac{2}{\delta}) \qquad (6)$$

Therefore, after seeing $s$ positive results, while processing $n$ points where $n$ is bounded by Equation 6 one can be confident that the clusters will be stable and the probability of successfully clustering a point is the expected value of the random variable $X$ divided by $n$ (the total number of points that we attempted to cluster).

These bounds can be used to drive our tracking algorithm *Tracking*, described in Figure 2. Essentially, the algorithm takes $n$ new points (where $n$ is given by the lower bound of Equation 6) and checks how many of them can be successfully clustered by FC, using the current set of clusters. (Recall that if a point has a MFI bigger than $\tau$, it is deemed an outlier.) If after attempting to cluster the $n$ points, one finds too many outliers (tested in Line 9, by comparing the successful count $r$, with the computed bound $s$, given by Equation 5), then we call this a *turning point* and proceed to redefine the clusters. This is done by throwing away all the information of the previous clusters and clustering the $n$ points of the current batch. Notice that after each iteration, the value of $p$ is re-estimated as the ratio of successfully clustered points divided by the total number of points tried.

## Experiments

We describe in this section the result of two experiments using our *tracking* algorithm. We performed the experiments in a Sun Ultra2 with 500 Mb. of RAM, running Solaris 2.5.

0. Initialize the count of successfully clustered points, i.e., $r = 0$
1.Given a batch $S$ of $n$ points, where $n$ is computed as the lower bound of Equation 6, using the estimated $p$ from the previous round of points
2.For each point in $S$:
    3. Use FC to cluster the point.
    4. If the point is not an outlier
        5. Increase the count of successfully clustered points, i.e., $r = r + 1$
6. Compute $s$ as the lower bound of Equation 5
7. If $r < s$, flag this batch of points $S$ as a turning point and use $S$ to find the new clusters.
8. Else re-estimate $p = r/n$

**Figure 2: *Tracking*: Algorithm to track cluster changes.**

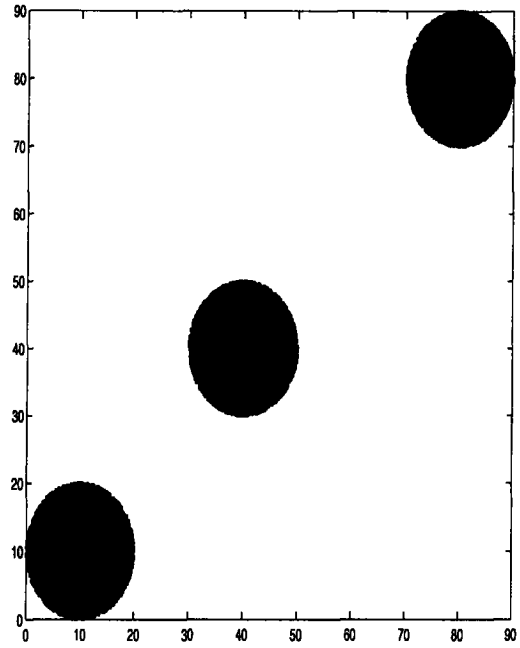


**Figure 3: Three-cluster data set used for scalability experiments.**

In Figure 3, we show the data we used for the first experiment, in a graphic form. (We show in this figure the whole set of points, but not how they were presented to the algorithm, which is explained in what follows.) The data set is composed by three kinds of points. The first kind, which appear at the beginning of the data set, are points located in the left bottom circle of Figure 3. The second kind of points, located after all the points of the first kind, correspond to points belonging to the middle circle of Figure 3. Finally, the third kind of points, belonging to the upper right circle, are located at the end of the data set. The data set is read in batches of points. The initial clustering of points (of the first kind) results in a single cluster, corresponding to the left-bottom circle. While points of the first kind are processed by the algorithm *Tracking*, no change of clusters is found. When points of the second kind start showing in the batch, the algorithm reports 3759 outliers in a batch of 4414 ($n$) points. Since $r = 4414 - 3759 = 655$ is much smaller than $s$ which has been computed at this point to be 3774, a turning point is flagged and the 4414 points are clustered anew, resulting in a single cluster corresponding to the middle circle. When points of the upper-right circle are tested in with the algorithm *Tracking*, the same phenomena occurs. The experiment was conducted with the values $\delta = 0.1$, $\epsilon = 0.05$ and it took 0.3 seconds to run.

The second experiment used data from the U.S. Historical Climatology Network (CDIA ), which contains (among other types of data) data sets with the average temperature per month, for several years measured in many meteorological stations throughout the United States. We chose the data for the years 1990 to 1994 for the state of Virginia for this experiment (the data comes from 19 stations throughout the state). We organized the data as follows. First we feed the algorithm with the data of the month of January for all the years 1990-1994, since (we were interested in finding how the average temperature changes throughout the months of the year, during those 5 years. Our clustering algorithm found initially a single cluster for points throughout the region in the month of January. This cluster contained 1,716 data points. Using $\delta = 0.15$, and $\epsilon = 0.1$, and with the estimate of $p = 0.9$ (given by the number of initial points that were successfully clustered), we get a window $n = 1055$, and a value of $s$, the minimum number of points that need to be clustered successfully, of 855. (Which means that if we find more than 1055-855 = 200 outliers, we will declare the need to re-cluster.) We proceeded to feed the data corresponding to the next month (February for the years 1990-1994) in chunks of 1055 points, always finding less than 200 outliers per window. With the March data, we found a window with more than 200 outliers and decided to re-cluster the data points (using only that window of data). After that, with the data corresponding to April, fed to

the algorithm in chunks of $n$ points ($p$ stays roughly the same, so $n$ and $s$ remain stable at 1055 and 255, respectively) we did not find any window with more than 200 outliers. The next window that prompts re-clustering comes within the May data (for which we reclustered). After that, re-clustering became necessary for windows in the months of July, October and December. The $\tau$ used throughout the algorithm was 0.001. The total running time was 1 second, and the total number of data points processed was 20,000.

## Conclusions

In this paper we have presented an algorithm to track changes in cluster models for evolving data sets. This problem becomes important as organizations accumulate new data and are interested in analyzing how the patterns in the data change over time. Our algorithm was able to track perfectly changes on the two datasets we experimented with.

Although the idea of using bounds (based on Chernoff's inequality) can be applied to other clustering algorithms (as long as they exhibit incremental behavior), Fractal Clustering is specially well suited for this, due to its incremental nature, the concise way in which the current clusters information is kept in main memory and the natural way in which the algorithm deals with noise (by usage of a simple threshold).

We plan to continue the evaluation of our algorithm, with larger datasets (for instance, the complete set of data found in (CDIA )) and other applications.

## References


Barbará, D., and Chen, P. 2000. Using the Fractal Dimension to Cluster Datasets. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA.*

Belussi, A., and Faloutsos, C. 1995. Estimating the Selectivity of Spatial Queries Using the 'Correlation' Fractal Dimension. In *Proceedings of the International Conference on Very Large Data Bases*, 299–310.

CDIA. U.S. Historical Climatology Network Data. `http://cdiac.esd.ornl.gov/epubs/ndp019/ushcn\_r3.html`.

Chernoff, H. 1952. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations. *Annals of Mathematical Statistics* 493–509.

Domingo, C.; Gavaldá, R.; and Watanabe, O. 1998. Practical Algorithms for Online Selection. In *Proceedings of the first International Conference on Discovery Science.*

Domingo, C.; Gavaldá, R.; and Watanabe, O. 2000. Adaptive Sampling Algorithms for Scaling Up Knowledge Discovery Algorithms. In *Proceedings of the second International Conference on Discovery Science.*

Domingos, P., and Hulten, G. 2000. Mining High-Speed Data Streams. In *Proceedings of the Sixth ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA.*

Faloutsos, C., and Gaede, V. 1996. Analysis of the Z-ordering Method Using the hausdorff Fractal Dimension. In *Proceedings of the International Conference on Very Large Data Bases*, 40–50.

Faloutsos, C., and Kamel, I. 1997. Relaxing the Uniformity and Independence Assumptions, Using the Concept of Fractal Dimensions. *Journal of Computer and System Sciences* 55(2):229–240.

Faloutsos, C.; Matias, Y.; and Silberschatz, A. 1996. Modeling Skewed Distributions Using Multifractals and the '80-20 law'. In *Proceedings of the International Conference on Very Large Data Bases*, 307–317.

Grassberger, P., and Procaccia, I. 1983. Characterization of Strange Attractors. *Physical Review Letters* 50(5):346–349.

Grassberger, P. 1983. Generalized Dimensions of Strange Attractors. *Physics Letters* 97A:227–230.

Liebovitch, L., and Toth, T. 1989. A Fast Algorithm to Determine Fractal Dimensions by Box Countig. *Physics Letters* 141A(8).

Lipton, R., and Naughton, J. 1995. Query Size Estimation by Adaptive Sampling. *Journal of Computer Systems Science* 18–25.

Lipton, R.; Naughton, J.; Schneider, D.; and Seshadri, S. 1993. Efficient Sampling Strategies for Relational Database Operations. *Theoretical Computer Science* 195–226.

Mandelbrot, B. 1983. *The Fractal Geometry of Nature.* New York: W.H. Freeman.

Menascé, D.; Almeida, V.; Fonseca, R.; and Mendes, M. 1999. A Methodology for Workload Characterizatoin for E-commerce Servers. In *Proceedings of the ACM Conference in Electronic Commerce, Denver, CO.*

Schroeder, M. 1991. *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise.* New York: W.H. Freeman.

Selim, S., and Ismail, M. 1984. K-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6(1).

Watanabe, O. 2000. Simple Sampling Techniques for Discovery Science. *IEICE Transactions on Information and Systems.*