

A Framework for Diagnosing Changes in Evolving Data Streams

Charu C. Aggarwal
IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532
charu@us.ibm.com

ABSTRACT

In recent years, the progress in hardware technology has made it possible for organizations to store and record large streams of transactional data. This results in databases which grow without limit at a rapid rate. This data can often show important changes in trends over time. In such cases, it is useful to understand, visualize and diagnose the evolution of these trends. When the data streams are fast and continuous, it becomes important to analyze and predict the trends quickly in online fashion. In this paper, we discuss the concept of velocity density estimation, a technique used to understand, visualize and determine trends in the evolution of fast data streams. We show how to use velocity density estimation in order to create both *temporal velocity profiles* and *spatial velocity profiles* at periodic instants in time. These profiles are then used in order to predict three kinds of data evolution: dissolution, coagulation and shift. Methods are proposed to visualize the changing data trends in a single online scan of the data stream, and a computational requirement which is linear in the number of data points. In addition, batch processing techniques are proposed in order to identify combinations of dimensions which show the greatest amount of global evolution. The techniques discussed in this paper can be easily extended to spatio-temporal data, changes in data snapshots at fixed instances in time, or any other data which has a temporal component during its evolution.

1. INTRODUCTION

With the large number of transactions which are recorded and stored by many organizations, the importance of being able to analyze trends in fast data streams has become very important in recent years. Typically, such databases are created by continuous activity over long periods of time, and are therefore databases which grow without limit. For example, even simple transactions of everyday life, such as paying by credit card or using the telephone are recorded

in an automated way using current hardware technology. The volume of such transactions may easily range in the millions on a daily basis. Often, the data may show important changes in the trends over time because of fundamental changes in the underlying phenomena. This process is referred to as *data evolution*. By understanding the nature of such changes, a user may be able to glean valuable insights into emerging trends in the underlying transactional or spatial activity. Therefore, it is useful to develop tools and techniques which would provide a visual and diagnostic overview of the key characteristics in the data which have changed over time in a fast and user-friendly way.

The problem of mining high speed data streams has been recognized to be an important one in recent years [1, 2, 6, 7, 18]. There is a considerable amount of work in the literature with a focus on incremental maintenance of models in the context of evolving data [5, 8, 17]. Recent work [10, 11] has discussed a general framework for quantifying the changes in evolving data characteristics in the context of several data mining problems and algorithms. The focus of our paper is different from and orthogonal to the work in [10, 11]. Specifically, the work in [10, 11] is focussed on the effects of evolution on data mining models and algorithms. This paper studies the problem of evolution in terms of understanding the *nature of the data changes* as opposed to finding whether a particular data mining model has become stale or not because of the underlying change. As we shall see, such a study of evolution provides the user with considerable understanding of the changes more directly in terms of the data attributes. Other recent work [1, 4, 12] discusses the problem of change detection and visualization for particular domains of data.

Since a data stream typically contains a large volume of high dimensional information, it is useful to characterize the underlying changes in a fast and user-friendly way. For example, what are the locations in the stream which show the greatest amount of change? How can the change be characterized in a way which is intuitively appealing to a user? This paper provides the user a generic tool to understand, visualize and diagnose the summary changes in data characteristics. This includes methods to visualize how the pattern of the data in various spatial locations has changed over time, or which combinations of dimensions show the greatest amount of change. We also discuss ways of providing specific diagnosis of localized phenomena in different spatial regions of the data. Such information can be used in order to understand the nature of the emerging trends in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGMOD 2003, June 9-12, 2003, San Diego, CA
Copyright 2003 ACM 1-58113-634-X/03/06 ...\$5.00.

the data.

There are several interesting applications of change diagnosis in evolving data streams. For example, in a supermarket or electronic commerce application, a stream of customer requests may consist of demographic or customer buying behavior attributes. It may often be desirable to pick those combinations of attributes in which there is large change in the level of transactional activity. The ability to track such information is a clear competitive advantage for a business, which can use it in order to anticipate and adapt to potential customers in the near future. For instance, if recent data shows an increasing trend in the number of customers with a certain combination of demographic characteristics, then the supermarket can anticipate the increased demands of the items usually bought by this group. Another example is a GIS application in which the spatial coordinates of a very large number of mobile objects are being tracked simultaneously. This can be modeled as a data stream in which the attributes correspond to the spatial coordinates of the objects. Even though the number of objects may be large, their overall motion may show systematic trends in many regions of the data. It is often useful to mine the (aggregate) spatial trends of these objects by finding the localized nature of the data re-organization.

When the data stream shows a high level of evolution, it is expected that the relative data concentrations at various spatial locations may change over time. We will capture such changes with the concept of velocity density which measures the rate of change of data concentration at a given spatial location over a user-defined time horizon. This density can be used in order to create two kinds of visual profiles of data evolution: the *temporal velocity profiles* and *spatial velocity profiles*. These profiles provide different perspectives on the nature of the underlying change. Since the results of this paper are relevant to high dimensional data mining problems as well, it is important to provide diagnosis capability for such cases. In order to deal with this class of problems effectively, we use velocity density estimation in order to pick the projections from high dimensional data in which the greatest changes in data characteristics have occurred. Such combinations of dimensions are very interesting, because it may often happen that there may be very little evolution in terms of either the entire set of dimensions or individual dimensions; yet some particular combinations of dimensions may show huge levels of evolution because of changes in the correlation structure of the data. A closely related problem is that of mining spatio-temporal or mobile data [14, 15, 16], for which it is useful to have the ability to diagnose aggregate changes in spatial characteristics over time. The results of this paper are equally valuable for providing such understanding.

Since data streams are fast and continuous over long periods of time, the data may quickly become stale for time-sensitive applications. In such cases, the usefulness of the evolution monitoring process also becomes time-critical. In order to achieve this goal, we mine the data streams in an online fashion; therefore, most techniques discussed in the paper need to examine a data point only once throughout the entire computation. To this effect, we discuss algorithms whose computational requirement per record in the stream are constant, and which do not require any re-scanning of the data. This is the most optimistic case that can be achieved by any algorithm for mining data streams.

This paper is organized as follows. In the remainder of this section, we formalize the contributions of this paper and provide a brief overview of some concepts from kernel density estimation which we will need in this paper. In section 2, we will introduce the velocity density estimation technique and show how to use it in order to construct the temporal velocity profile. In section 3, we will discuss how the temporal velocity profiles can be used in order to generate the spatial velocity profiles. These profiles can be used in order to provide a visual understanding of the spatial re-organization of the data. Extensions to high dimensional data are discussed in section 4. In section 5, we discuss how the temporal and spatial profiles may be used to provide a more concrete diagnosis of the emerging trends in localized regions. In section 6, we present the empirical results which show interesting evolution behavior for different data sets. We also show the performance efficiency of the technique. The conclusion and summary is presented in section 7.

1.1 Contributions of this paper

This paper provides a framework for effective diagnosis of multidimensional data streams with the use of a concept called velocity density estimation. This definition is both intuitively appealing and can be computed efficiently for a fast data stream. It is also possible to derive a good visual perspective of the nature of data evolution with the use of two profiles: a spatial velocity profile, and a temporal velocity profile. These profiles are also used in order to diagnose important trends in the data. For example, spatial re-organizations in localized regions in the data can be detected by the use of this method. For the case of high dimensional data, we propose batch-processing methods in order to pick projections from the data set in which the greatest amount of evolution has occurred. This is especially useful in cases when all dimensions do not show a similar amount of change or when the underlying correlation structure of the data has changed only over some particular subsets of dimensions. All the methods of this paper can be directly extended in order to diagnose changes between two snapshots of data (or two static data sets). Most of the algorithms of this paper are designed to work efficiently with one scan of the data so that the profiles can be constructed in online fashion. Thus, the technique is very scalable, a factor which is important for mining very high speed data streams.

1.2 Kernel Density Estimation Overview

The idea in kernel density estimation [13] is to provide a continuous estimate of the density of the data at a given point. The value of the density at a given point is estimated as the sum of the smoothed values of kernel functions $K'_h(\cdot)$ associated with each point in the data set. Each kernel function is associated with a kernel width h which determines the level of smoothing created by the function. The kernel estimation $\bar{f}(x)$ based on n data points and kernel function $K'_h(\cdot)$ is defined as follows:

$$\bar{f}(x) = (1/n) \cdot \sum_{i=1}^n K'_h(x - X_i) \quad (1)$$

Thus, each discrete point X_i in the data set is replaced by a continuous function $K'_h(\cdot)$ which peaks at X_i and has a variance which is determined by the smoothing parameter h . An example of such a distribution would be a gaussian

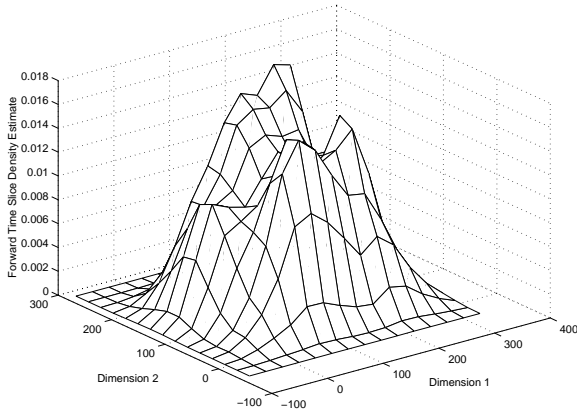


Figure 1: The Forward Time Slice Density Estimate

kernel with width h .

$$K'_h(x - X_i) = (1/\sqrt{2\pi} \cdot h) \cdot e^{-(x-X_i)^2/(2h^2)} \quad (2)$$

The overall effect of kernel density estimation is to convert the (discrete) data set into a continuous density estimate by replacing each data point with a smoothed bump, whose width is determined by h . The density distribution at a given coordinate is equal to the sum of the contributions of all the bumps represented by the data points. The result is a continuous distribution in which the random artifacts are suppressed and the density behavior provides a global overview of the dense as well as sparsely populated regions of the data. The estimation error depends upon the kernel width h which is chosen in a data driven manner. It has been shown [13] that for most smooth functions $K'_h(\cdot)$, when the number of data points goes to infinity, the estimator $\bar{f}(x)$ asymptotically converges to the true density function $f(x)$, provided that the width h is chosen appropriately. For the d -dimensional case, the kernel function is chosen to be the product of d identical kernels $K_i(\cdot)$, each with its own smoothing parameter h_i .

2. VELOCITY DENSITY ESTIMATION

The idea in velocity density estimation is to estimate the rate at which the changes in the data density are occurring at each spatial location based on some user-defined temporal window h_t . In general, when a user is faced with a massive data stream, there are several interesting questions which arise: for example, what are the spatial locations at which the data is increasing, reducing or shifting to? What is the level of changes in the data characteristics occurring at different spatial locations? Intuitively, the temporal window h_t is associated with the time horizon over which the rate of change is measured. Thus, if h_t is chosen to be large then the velocity density estimation technique provides long term trends, whereas if h_t is chosen to be small then the trends are relatively short term. This provides the user flexibility in analyzing the changes in the data over different kinds of time horizons. In addition, we have a spatial smoothing vector h_s whose function is quite similar to the standard spatial smoothing vector which is used in kernel density estimation.

Let t be the current instant and S be the set of data points which have arrived in the time window $(t-h_t, t)$. We intend to estimate the rate of increase in density at spatial

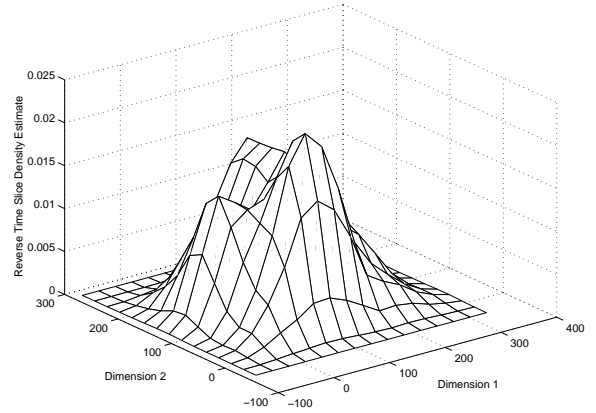


Figure 2: The Reverse Time Slice Density Estimate

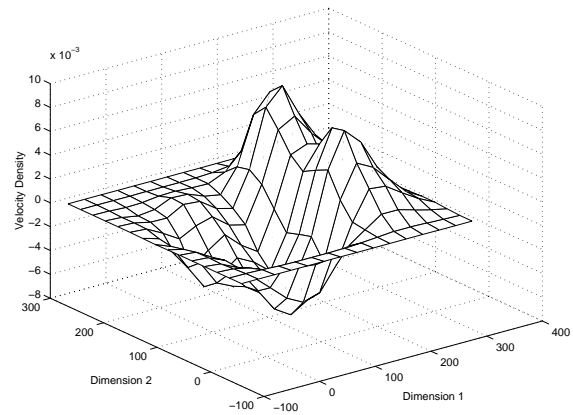


Figure 3: The Temporal Velocity Profile

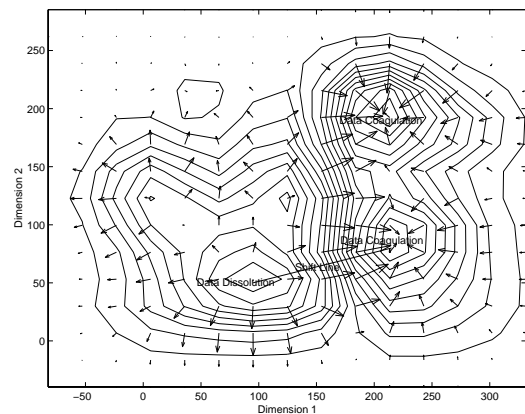


Figure 4: The Spatial Velocity Profile

location X and time t by using two sets of estimates: the *forward time slice density estimate* and the *reverse time slice density estimate*. Intuitively, the forward time slice estimate measures the density function for all spatial locations at a given time t based on the set of data points which have arrived in the *past* time window $(t - h_t, t)$. Similarly, the reverse time slice estimate measures the density function at a given time t based on the set of data points which will arrive in the *future* time window $(t, t + h_t)$. Let us assume that the i th data point in S is denoted by (X_i, t_i) , where i varies from 1 to $|S|$. Then, the forward time slice estimate $F_{(h_s, h_t)}(X, t)$ of the set S at the spatial location X and time t is given by:

$$F_{(h_s, h_t)}(X, t) = C_f \cdot \sum_{i=1}^{|S|} K_{(h_s, h_t)}(X - X_i, t - t_i) \quad (3)$$

Here $K_{(h_s, h_t)}(\cdot, \cdot)$ is a spatio-temporal kernel smoothing function, h_s is the spatial kernel vector, and h_t is temporal kernel width. The kernel function $K_{(h_s, h_t)}(X - X_i, t - t_i)$ is a smooth distribution which decreases with increasing value of $t - t_i$. The value of C_f is a suitably chosen normalization constant, so that the entire density over the spatial plane is one unit. This is done, because our purpose of calculating the densities at the time slices is to compute the *relative* variations in the density over the different spatial locations. Thus, C_f is chosen such that we have:

$$\int_{\text{All } X} F_{(h_s, h_t)}(X, t) \delta X = 1 \quad (4)$$

The reverse time slice density estimate is also calculated in a somewhat different way to the forward time slice density estimate. We assume that the set of points which have arrived in the time interval $(t, t + h_t)$ is given by U . As before, the value of C_r is chosen as a normalization constant. Correspondingly, we define the value of the reverse time slice density estimate $R_{(h_s, h_t)}(X, t)$ as follows:

$$R_{(h_s, h_t)}(X, t) = C_r \cdot \sum_{i=1}^{|U|} K_{(h_s, h_t)}(X - X_i, t_i - t) \quad (5)$$

Note that in this case, we are using $t_i - t$ in the argument instead of $t - t_i$. Thus, the reverse time-slice density in the interval $(t, t + h_t)$ would be exactly the same as the forward time slice density if we assumed that time was reversed and the data stream arrived in reverse order, starting at $t + h_t$ and ending at t . Examples of the forward and reverse density profiles are illustrated in Figures 1 and 2 respectively.

For a given spatial location X and time T , let us examine the nature of the functions $F_{(h_s, h_t)}(X, T)$ and $R_{(h_s, h_t)}(X, T - h_t)$. Note that both functions are almost exactly the same, and use the same data points from the interval $(T - h_t, T)$, except that one has been calculated assuming time runs forward, whereas the other has been calculated assuming that the time runs in reverse. Furthermore, the volumes under each of these curves, when measured over all spatial locations X is equal to one unit because of the normalization. Correspondingly, the density profiles at a given spatial location X would be different between the two depending upon how the relative trends have changed in the interval $(T - h_t, T)$. We define the velocity density $V_{(h_s, h_t)}(X, T)$ at

spatial location X and time T as follows:

$$V_{(h_s, h_t)}(X, T) = \frac{F_{(h_s, h_t)}(X, T) - R_{(h_s, h_t)}(X, T - h_t)}{h_t} \quad (6)$$

Note that the velocity density is positive at a given spatial location X , if in the interval $(T - h_t, T)$ a greater number of points which are closer to X have arrived at the end of the interval. On the other hand, when a greater number of points which are closer to X are at the beginning of the interval $(T - h_t, T)$, then the velocity density is negative at the spatial location. If the trends have largely remained unchanged over the interval, then the velocity density at the location X will be almost zero. Thus, the velocity density at a given point provides an intuitive understanding of the rate of change of the density at that spatial location over the time horizon $(T - h_t, T)$. A global overview of the rate of changes of densities at different points specific to the time T is referred to as the *temporal velocity profile*. An example of a temporal velocity profile (which will be discussed in a greater detail in the empirical section) is illustrated in Figure 3. Note that the total volume which is trapped between the temporal velocity profile curve and the spatial axis plane is at most $2/h_t$, since the total volume under each of the forward and reverse time slice density curves is exactly one unit. The larger this volume, the greater the total amount of evolution that has occurred. In a later section, we will discuss how this volume may be used in order to provide evolution rates for a data stream which are intuitively comprehensible to a user.

2.1 Choice of Kernel Function

The spatio-temporal kernel function is a time-factored version of the spatial kernel. In other words, if h_s be the spatial smoothing vector, and h_t be the temporal smoothing parameter (defined by user time-horizon), then the spatio-temporal kernel function is defined as follows:

$$K_{(h_s, h_t)}(X, t) = (1 - t/h_t) \cdot K'_{h_s}(X) \quad (7)$$

This kernel function is only defined for values of t in the range $(0, h_t)$. The gaussian spatial kernel function $K'_{h_s}(\cdot)$ was used because of its well known effectiveness [13]. Specifically, $K'_{h_s}(\cdot)$ is the product of d identical gaussian kernel functions, and $h_s = (h_s^1, \dots, h_s^d)$, where h_s^i is the smoothing parameter for dimension i .

2.2 Physical Significance of Velocity Density

The above choice of kernel function also leads to an interesting physical significance of the velocity density computation. Let us explore a prototypical data stream in which the points arrive at a constant rate of ν per second throughout the time interval $(T - h_t, T)$. Further, let the spatial density function of all points arriving in the time interval $(\theta, \theta + d\theta) \in (T - h_t, T)$ be given by $g(X, \theta)$. Let us also assume that for each *specific value* of X , $\partial g(X, \theta)/\partial \theta$ is constant over the interval $(T - h_t, T)$ and is equal to $\mu(X)$. Then, we can show that in the *asymptotic* case (when ν is arbitrarily high), the velocity density $V_{(h_s, h_t)}(X, T)$ at the spatial location X and time T is proportional to $\mu(X) = \partial g(X, \theta)/\partial \theta$.

For $t \in (0, h_t)$, let us consider the small time-interval $(T - h_t + t, T - h_t + t + dt)$. Then, the number of data points which have arrived in this interval is proportional to $\nu \cdot dt$ which is constant over any interval of length dt in

$(T - h_t, T)$. We know that the contribution of the points in this interval to $V_{(h_s, h_t)}(X, T)$ is proportional to the difference between the corresponding contributions to the forward and reverse time slice densities. We further note that the spatio-temporal kernel function is a time-factored version of the spatial kernel function. Therefore, we can assume that if we used only the points in the interval $(T - h_t + t, T - h_t + t + dt)$ to estimate the forward and reverse time slice densities, then the results of [13] would indicate that these contributions asymptotically converge in proportionality to $g(X, T - h_t + t) \cdot (t/h_t)$ and $g(X, T - h_t + t) \cdot (1 - t/h_t)$ respectively. This means that the contribution difference converges to a value which is proportional to $([g(X, T - h_t + t)] \cdot (t/h_t) - [g(X, T - h_t + t)](1 - t/h_t)) \cdot (\nu \cdot dt)$. Therefore, the value of $V_{(h_s, h_t)}(X, T)$, when summed over all the contributions of length dt in the interval $(T - h_t, T)$ is given by:

$$V_{(h_s, h_t)}(X, T) \propto \int_{t=0}^{h_t} \{ [g(X, T - h_t + t)] \cdot (t/h_t) - [g(X, T - h_t + t)] \cdot (1 - t/h_t) \} dt$$

On expanding the value of $g(X, T - h_t + t) = g(X, T - h_t) + t \cdot \mu(X)$ in the above result and regrouping, we get:

$$V_{(h_s, h_t)}(X, T) \propto [g(X, T - h_t)] \int_{t=0}^{h_t} (2t/h_t - 1) dt + [\mu(X)] \int_{t=0}^{h_t} (2t^2/h_t - t) dt$$

Note that the first term of the above expression is zero, which means that the velocity density at spatial location X is proportional to the value of $\mu(X)$ during the interval $(T - h_t, T)$. The above result is *not* true when the stream rate ν changes with time; this is because the velocity density is supposed to measure *relative* changes of densities over the different spatial locations.

2.3 Special Case for Static Snapshots

An interesting special case of the method is when we would like to detect the changes between two static snapshots of data at a time interval of h_t . Thus, in this case we assume that the first set of points all arrived at time $T - h_t$, whereas the second set arrived at time T . When the method is applied to this special case, then the velocity density is proportional to the difference in the *spatial* kernel densities of the two data sets. It is clear how the magnitude and sign of the corresponding density at a given point would provide an insight into the changes which have occurred between the two data sets.

2.4 Effective Implementation

In order to construct the velocity profiles, we need a way of picking a reasonable set of coordinates for plotting the data. The discretized version of the velocity density is used for this purpose. We pick a total of β coordinates along each dimension. For a 2-dimensional system, this corresponds to β^2 spatial coordinates. The higher the value of β , the better the resolution for the visualization system; on the other hand, the computational requirements are greater. First, the velocity density is calculated at each of these β^2 points. The temporal velocity profile can be calculated by a simple $O(\beta^2)$ additive operations per data point.

Let us say that it has been decided to calculate the temporal velocity profile in the interval $(t - h_t, t)$. Then, for each coordinate X_g in the grid, we maintain two sets of counters (corresponding to forward and reverse density counters)

which are updated as each point in the data stream is received. When a data point X_i is received at time t_i , then we add the value $K_{(h_s, h_t)}(X_g - X_i, t - t_i)$ to the forward density counter, and the value $K_{(h_s, h_t)}(X_g - X_i, t_i - (t - h_t))$ to the reverse density counter for X_g .

At the end of time t , the values computed for each coordinate at the grid need to be normalized. The process of normalization is the same for either the forward or the reverse density profiles. In each case, we sum up the total value in all the β^2 counters, and divide each counter by this total. Thus, for the normalized coordinates the sum of the values over all the β^2 coordinates will be equal to 1. This is analogous¹ to the requirement expressed in Equation 4.

Successive sets of temporal profiles are generated at user-defined time-intervals of h_t . In order to ensure online computation, the smoothing parameter vector h_s for the time-interval $(T - h_t, T)$ must be available at time $T - h_t$, as soon as the first data point of that interval is scheduled to arrive. Therefore, we need a way of estimating this vector using the data from past intervals. In order to generate the velocity density for the interval $(T - h_t, T)$, the spatial kernel smoothing vector h_s is determined using the Silverman's approximation rule² [13] for gaussian kernels on the set of data points which arrived in the interval $(T - 2h_t, T - h_t)$.

3. SPATIAL VELOCITY PROFILES

In the previous section, we discussed how to calculate the temporal velocity profiles which illustrate the rate of change of density at each fixed spatial location. Even better insight can be obtained by examining the nature of the spatial velocity profiles, which provide an insight into how the data is shifting. For each spatial point, we would like to compute the directions of movements of the data at a given instant. The motivation in developing a spatial velocity profile is to give a user a *spatial* overview of the re-organizations in relative data density at different points. This brings to us the following interesting problem: given the temporal velocity profile, how do we calculate the directions in which the data is shifting? In order to understand this, we observe that when data is shifting from one point to another, the source of the shift shows a reduction in the density, whereas the destination of the shift shows an increase in the density. Therefore, there is an increasing density gradient from the source to the destination. The exact direction of the (aggregate) shift is determined by finding the direction in which the gradient is the largest.

Let us define an ϵ -perturbation along the i th dimension by $\bar{\epsilon}_i = \epsilon \cdot \bar{e}_i$, where \bar{e}_i is the unit vector along the i th dimension. For a given spatial location X , we first compute the velocity gradient along each of the i dimensions. We denote the velocity gradient along the i th dimension by $\Delta v_i(X, t)$ for spatial location X and time t . This value is computed by subtracting the density at spatial location X from the density at $X + \bar{\epsilon}_i$ (ϵ -perturbation along the i th dimension), and dividing the result by ϵ . The smaller the value of ϵ , the

¹To be precise, we assume (without loss of generality) that the area of each elementary grid rectangle is one unit.

²According to Silverman's approximation rule, the smoothing parameter for a data set with n points and standard deviation σ is given by $1.06 \cdot \sigma \cdot n^{-1/5}$. For the d -dimensional case, the smoothing parameter along each dimension is determined independently using the corresponding dimension-specific standard deviation.

better the approximation. Therefore, we have:

$$\Delta v_i(X, t) = \lim_{\epsilon \rightarrow 0} \frac{V_{(h_s, h_t)}(X + \overline{\epsilon}_i, t) - V_{(h_s, h_t)}(X, t)}{\epsilon} \quad (8)$$

The value of $\Delta v_i(X, t)$ is negative when the velocity density decreases with increasing value of the i th coordinate of spatial location X . The gradient $\overline{\Delta v}(X, t)$ is given by $(\Delta v_1(X, t) \dots \Delta v_d(X, t))$. This vector gives the spatial gradient at a given grid point *both* in terms of direction and magnitude. The spatial velocity profile is illustrated by creating a spatial plot which illustrates the directions of the data shifts at different grid points by directed markers which mirror these gradients both in terms of directions and magnitude. An example of a spatial velocity profile is illustrated in Figure 4. More details about this plot will be discussed in the empirical section.

3.1 Effective Implementation

In order to provide visualization capability, we determine the spatial profiles at a specific set of grid points. Let X_g be such a grid point. Then the value of $\Delta v_i(X_g, t)$ is calculated using the perturbed velocity densities along the i th dimension. Therefore, for each grid point, we maintain an additional d perturbed values, one for each dimension. Specifically, for a grid point X_g , the i th perturbed value is the velocity density at the point $X_g + \overline{\epsilon}_i$. Thus, during the on-line scan of the data, we need to maintain an additional d counters for each grid point at which these densities are stored. At the end of the scan, the spatial profiles can be computed by calculating each of the d components of the spatial velocity as illustrated in Equation 8. Note that this extra effort at the end of the velocity density calculation is (asymptotically) small for a rapid data stream, since it is independent of the number of records in the stream. The actual value of the perturbation ϵ should be chosen as small as possible subject to the numerical rounding errors created by choosing ϵ near the computer numerical precision. For a practical implementation, we consistently chose ϵ to be 1% of the distance between two adjacent grid points.

4. EXTENSION TO HIGH DIMENSIONAL CASE

Although the velocity density can be calculated for arbitrary dimensions, the visualization of the spatial and temporal profiles is inherently 2-dimensional. In this section, we discuss how to utilize the results of the velocity density estimation techniques in order to make significant diagnosis about higher dimensional problems. An important fact to be noted about high dimensional problems is that the data may not show change on *all* the sets of the dimensions, but on some particular combinations; therefore a significant amount of useful information is hidden in finding the combinations of dimensions in which the greatest amount of change has occurred. It is clear that that if the change is measured over the entire set of dimensions simultaneously, its rate may be significantly diluted by those combinations of dimensions in which the evolution rate is insignificant. Furthermore, since high dimensional data is sparse, it is not very interesting to measure evolution rates of sparse data concentrations. In fact, is not even possible to estimate densities accurately with increasing dimensionality [13]. Therefore, we will show how to use the concept of velocity density in

order to pick interesting sets of features which show considerable evolution.

Note that the temporal velocity profile gives the *rate* of change at a given spatial location X . Therefore, by integrating the value of the velocity density over the entire spatial area, we can obtain the total rate of change over the entire spatial area. In other words, if $E_{(h_s, h_t)}(t)$ be the total evolution in the period $(t - h_t, t)$, then we have:

$$E_{(h_s, h_t)}(t) = h_t \int_{\text{all } X} |V_{(h_s, h_t)}(X, t)| \delta X$$

We shall henceforth refer to the above value as the *global evolution coefficient* of the data set. It is important to note that we have to use the modulus of the velocity, since we are looking at the total volume of change, rather than the positivity or the negativity of the change. (If we did not use the modulus, this integral would be equal to zero.) Note that the evolution coefficient is equal to the (scaled) volume between the velocity density curve and spatial axis plane. Because of the way in which the forward and reverse time slice densities are defined, this value always lies in the range $(0, 2)$. Intuitively, the evolution coefficient measures the total volume of the evolution in the time horizon $(t - h_t, t)$.

Since we have consistently computed the velocity densities at grid-discretized coordinate points, the corresponding grid-discretized approximation value of $E_{(h_s, h_t)}(t)$ is given by multiplying h_t with the sum of *absolute* values at all the different grid-coordinates.

It is also easy to see that it is possible to calculate the evolution coefficients of particular projections of the data by using only the corresponding sets of dimensions in the density calculations. Now consider a d -dimensional data mining problem in which we wish to find visual representations of a small number l of 2-dimensional projections in which the greatest change in data characteristics has occurred. Let $\gamma = \beta^2$ be the number of coordinates in each grid which are maintained at any given instant of time. Then, for each of the $\binom{d}{2}$ 2-dimensional combinations of attributes, we maintain separate sets of γ counters. For each of these sets, the velocity density is calculated by using the computation only in the corresponding projection of the data. Once the velocity density has been calculated at all the grid-points, we can calculate the grid-discretized approximation of the global evolution coefficient for a given projection. The l projections which have the highest evolution coefficient are the ones in which visual representations of the spatial and temporal profiles can be provided.

It is also of value to examine the rate of change in projections with dimensionality larger than 2. We have discussed earlier that it may often not be interesting to measure evolution behavior on the entire set of dimensions at one time. The reverse is also true; even though the data stream may not evolve much when projected onto one or two dimensions, there may be significant changes in particular sets of k -dimensional combinations. These sets of k -dimensional combinations provide useful information in terms of how the nature of the correlations have changed over time. (For example, the beginning of the data stream could correspond to records in which older people earn more than younger people, whereas the end of the data stream could correspond to the reverse.) Such information about lower dimensional projections is useful in order to obtain an idea of the exact nature of the trends in the data.

In order to find the sets of k -dimensional combinations in which the greatest change has occurred, we set a threshold

```

Procedure FindEvolvingProjections(Coefficient: min-evol,
MaxDimensionality: maxdim);
begin
 $\mathcal{C}_1 = \mathcal{S} = \{1, \dots, d\}$ ;  $\mathcal{F} = \text{null}$ ;  $i = 1$ ;
while ( $i \leq \text{maxdim}$ ) and ( $\mathcal{C}_i$  is non-empty) do
  begin
 $\mathcal{C}_{i+1} = \mathcal{C}_i(\text{pairwise concatenate})\mathcal{S}$ ;
  Prune any projection from  $\mathcal{C}_{i+1}$ , if any of its
  subsets are not present in  $\mathcal{C}_i$ ;
  Set up grid counters for each projection in the set
   $\mathcal{C}_{i+1}$  by discretizing each dimension into  $\beta$  ranges;
  Scan the data set once in order to compute the velocity
  densities at the corresponding grid-points of each
  projection in  $\mathcal{C}_{i+1}$ ;
  Compute glob. evol. coeff. of each projection in  $\mathcal{C}_{i+1}$ ;
  Determine projections  $\mathcal{Q}$  in  $\mathcal{C}_{i+1}$  which have a global
  evolution coefficient greater than min-evol;
 $\mathcal{F} = \mathcal{F} \cup \mathcal{Q}$ ;  $\mathcal{C}_{i+1} = \mathcal{C}_{i+1} - \mathcal{Q}$ ;  $i = i + 1$ ;
  end;
end;

```

Figure 5: Determining Evolving Projections

which we refer to as *min-evol*. The global evolution coefficient must be greater than this threshold for a projection to be considered significant. Such a projection is said to be *highly evolving*. From the perspective of a user, it is much easier to understand and assimilate the combination of a small number of dimensions. Therefore, we concentrate on finding the sets of lowest dimensional projections in which are highly evolving. A projection is said to be a *minimal evolving projection*, when its global evolution coefficient is greater than *min-evol* and no lower dimensional subspace of that projection is highly evolving. Although there may be a large number of highly evolving projections, the set of minimal evolving projections is likely to be small and easily presentable to a user. We will see that it is also computationally more efficient to find the set of minimal evolving projections by using pruning techniques. Note that any algorithm which tries to find all sets of projections in a single online scan of the data cannot prune any of the 2^d projections a priori, since it is not possible to make any evolution-coefficient estimates for any projection without scanning the data at least once. Such an algorithm may be feasible for reasonably small values of the dimensionality d , but becomes rapidly unwieldy with increasing dimensionality. Therefore, for the particular case of finding k -dimensional projections, we deviate from our general emphasis on developing only online algorithms. This is the only feature of this paper which requires offline computation and is done as a necessity because the computational requirements in one scan are untenable for *any* algorithm. We include this offline feature, since it may still be useful for many spatio-temporal data sets which can be loaded onto the disk for more detailed examination and analysis.

The algorithm for determination of minimum evolving projections derives its overall framework from the *Apriori* algorithm [3]. In Figure 5, we have outlined the basic algorithm for finding all the set of minimal evolving projections. For a d -dimensional data set, the method requires at most $d - 1$ scans over the data. The algorithm uses the minimality property in order to effectively prune many of the combinations of dimensions. The input to the algorithm consists of a minimum evolution threshold (denoted by *min-*

evol), and the maximum dimensionality of the projections (denoted by *maxdim*). A roll-up technique is employed in order to generate the combination of dimensions in which the set \mathcal{C}_i represents a set of i -dimensional *non-minimal candidate* projections. A projection P is said to be a non-minimal candidate, if and only if no subspace of P is highly evolving. At the beginning of the i th iteration, the set \mathcal{C}_{i+1} is defined as the set of non-minimal candidates; i.e. it is the set of all $(i + 1)$ -dimensional projections so that for each projection in it, no *proper* subspace³ of any projection in it has an evolution coefficient greater than *min-evol*. At the end of the iteration, the set of highly evolving $(i + 1)$ -dimensional projections are also removed from \mathcal{C}_{i+1} , so that \mathcal{C}_{i+1} is the set of all $(i + 1)$ -dimensional projections such that no subspace of any projection in it (including itself) is highly evolving. In each iteration, the candidate set \mathcal{C}_{i+1} is generated using the previous set \mathcal{C}_i , and the 1-dimensional singletons \mathcal{S} . More specifically, each element in \mathcal{C}_{i+1} is created by concatenating a projection in \mathcal{C}_i with a singleton in \mathcal{S} . The (*pairwise concatenate*) symbol in Figure 5 denotes this operation. Note that as the value of i increases a randomly chosen i -dimensional projection becomes increasingly unlikely to be a member of \mathcal{C}_i , since it is quite likely that at least one of its lower dimensional subspaces may be a highly evolving combination. How to (efficiently) check whether a given projection is non-minimal?

To do so, we use the non-minimality of \mathcal{C}_i inductively. If \mathcal{C}_i contains only non-minimal projections, then a projection in \mathcal{C}_{i+1} is non-minimal if and only if all of its i -dimensional subspaces are in \mathcal{C}_i . This is because if any proper subspace of a projection in \mathcal{C}_i satisfies the evolutionary threshold requirement, then such a projection would not itself be present in \mathcal{C}_i . Thus, we only need to check whether a 1-dimensional reduction from a projection in \mathcal{C}_{i+1} is present in \mathcal{C}_i , but we do not need to check lower dimensional subspaces. Thus, the pruning condition is similar to that discussed in [3] for association rule mining. The set of projections in \mathcal{C}_{i+1} are then *validated* by a scan over the data in order to check whether the evolution coefficient of each projection $P \in \mathcal{C}_{i+1}$ is higher than the threshold *min-evol*. Those projections \mathcal{Q} with evolution coefficient which is greater than *min-evol* are retained in each iteration and added to set of minimal evolving projections \mathcal{F} . Subsequently, \mathcal{Q} is removed from \mathcal{C}_{i+1} in order to ensure that the set of candidates \mathcal{C}_{i+2} generated in the next iteration do not contain any supersets of highly evolving projections from \mathcal{C}_{i+1} . The process is continued until either the candidate set is empty or projections of maximum dimensionality *maxdim* have been found. The set \mathcal{F} is returned at termination of the algorithm.

5. CHARACTERIZING THE DATA EVOLUTION

So far, we have discussed methods for effective visualization of the changes occurring in the data stream. An additional useful ability is to be able to concisely diagnose specific trends in given spatial locations. For example, a user may wish to know particular spatial locations in the data at which the data is being reduced, those at which the data is increasing, and those from where the data is shifting to other locations. In order to provide further insight, we introduce the following definitions:

³A proper subspace of X is any subspace except X .

DEFINITION 5.1. A data coagulation for time slice t and user defined threshold $min-coag$ is defined to be a connected region \mathcal{R} in the data space, so that for each point $X \in \mathcal{R}$, we have $V_{(h_s, h_t)}(X, t) > min-coag > 0$.

Thus, a data coagulation is a connected region in the data which has velocity density larger than a user-defined noise threshold of $min-coag$. In terms of the temporal velocity profile, these are the connected regions in the data with elevations larger than $min-coag$. Note that there may be multiple such elevated regions in the data, each of which may be disconnected from one another. Each such region is a separate area of data coagulation, since they cannot be connected by a continuous path above the noise threshold. For each such elevated region, we would also have a local peak, which represents the highest density in that locality.

DEFINITION 5.2. The epicenter of a data coagulation \mathcal{R} at time slice t is defined to be a spatial location X^* such that $X^* \in \mathcal{R}$ and for any $X \in \mathcal{R}$, we have $V_{(h_s, h_t)}(X, t) \leq V_{(h_s, h_t)}(X^*, t)$.

As with the other results of this paper, we use the grid discretization to provide a good approximation of the different shift regions and epicenters. We shall call the smallest rectangular grid area created by the grid discretization to be an elementary bounding rectangle. In the first step, we find all the elementary bounding rectangles so that all four corners of this bounding rectangle have a velocity density which is larger than $min-coag$. In the next step, we create connected components out of all such rectangles found. Two rectangles are said to be connected, if they share at least one common boundary. The entire region thus found is said to be a region of coagulation.

Once the regions of coagulation have been determined, the epicenter is found in a straightforward way by finding the grid point inside the coagulation region whose density is the highest. This is used as the approximation to the local peak of the temporal velocity profile within that region. The concept of data dissolution and its epicenter are similarly defined as follows:

DEFINITION 5.3. A data dissolution for time slice t and user defined threshold $min-dissol$ is defined to be a connected region \mathcal{R} in the data space, so that for each point $X \in \mathcal{R}$, we have $V_{(h_s, h_t)}(X, t) < -min-dissol < 0$.

Just as a data coagulation refers to a connected elevation in the temporal velocity profile, a data dissolution refers to a connected valley in the temporal velocity profile. Correspondingly, we define the epicenter of a data dissolution as follows:

DEFINITION 5.4. The epicenter of a data dissolution \mathcal{R} at time slice t is defined to be a spatial location X^* such that $X^* \in \mathcal{R}$ and for any $X \in \mathcal{R}$, we have $V_{(h_s, h_t)}(X, t) \geq V_{(h_s, h_t)}(X^*, t)$.

A region of data dissolution and its epicenter is calculated in an exactly analogous way to the epicenter of a data coagulation. It now remains to discuss how significant shifts in the data can be detected. Many of the epicenters of coagulation and dissolution are connected in a way which results in a funneling of the data from the epicenters of dissolution to the epicenters of coagulation. When this happens, it is clear that the two phenomena of dissolution and coagulation are

connected to one another. We refer to such a phenomenon as a *global data shift*. The detection of such shifts can be useful in many problems involving mobile objects. How to find whether a pair of epicenters are connected in this way?

In order to detect such a phenomenon we use the intuition derived from the use of the spatial velocity profiles. Let us consider a directed line drawn from an epicenter to data dissolution to an epicenter of data coagulation. In order for this directed line to be indicative of a global data shift, the spatial velocity profile should be such that the directions of a localized shifts along each of the points in this directed line should be in roughly in the same direction as the line itself. If at any point on this directed line, the direction of the localized shift is in an opposite direction, then it is clear that these two epicenters are disconnected from one another. In order to facilitate further discussion, we will refer to the line connecting two epicenters as a *potential shift line*.

Recall that the spatial velocity profiles provide an idea of the spatial movements of the data over time. In order to calculate the nature of the data shift, we would need to calculate the projection of the spatial velocity profiles along this potential shift line. In order to do so without scanning the data again, we use the grid points which are closest to this shift line in order to obtain an approximation of the shift velocities at various points along this line. The first step is to find all the elementary rectangles which are intersected by the shift line. Once these rectangles have been found we determine the grid points corresponding to the corners of these rectangles. These are the grid points at which the spatial velocity profiles are examined.

Let the set of n grid points thus discovered be denoted by $Y_1 \dots Y_n$. Then the corresponding spatial velocities at these grid points at time slice t are $\Delta v(Y_1, t) \dots \Delta v(Y_n, t)$. Let $\bar{\mathcal{L}}$ be the unit vector in the direction of the shift line. We assume that this vector is directed from the region of dissolution to the area of coagulation. Then the projections of the spatial velocities in the direction of the shift line are given by $\bar{\mathcal{L}} \cdot \Delta v(Y_1, t) \dots \bar{\mathcal{L}} \cdot \Delta v(Y_n, t)$. We shall refer to these values as $p_1 \dots p_n$ respectively. For a shift line to expose an actual movement of the data, the values of $p_1 \dots p_n$ must all be substantially positive. In order to quantify this notion, we introduce a user-defined parameter called *min-vel*. A potential shift line is said to be a valid shift when each of values $p_1 \dots p_n$ is larger than *min-vel*.

Thus, in order to determine all the possible data shifts, we first find all coagulation and dissolution epicenters for user-defined parameters *min-coag* and *min-dissol* respectively. Then we find all the potential shift lines by connecting each dissolution epicenter to a coagulation epicenter. For each such shift line, we find the grid points which are closest to it using the criteria discussed above. Finally, for each of these grid points, we determine the projection of the corresponding shift velocities along this line and check whether each of them is at least *min-vel*. If so, then this direction is reported as a valid shift line.

6. EMPIRICAL RESULTS

In order to test the methods discussed in this paper, we used a combination of real and synthetic data sets. The specific measures on which we tested the method were the following: (1) Examples of interesting spatial velocity and temporal velocity profiles created by the algorithm. (2) The

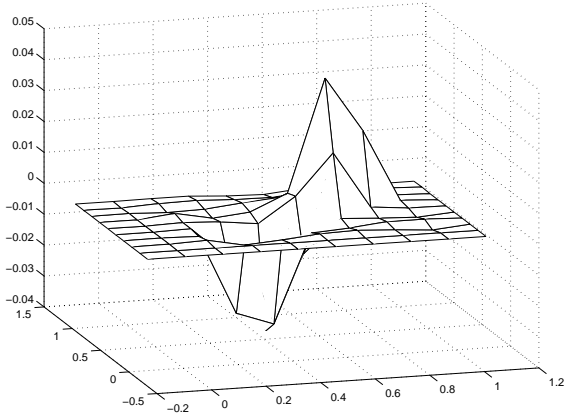


Figure 6: The Temporal Velocity Profile (S-Stream)

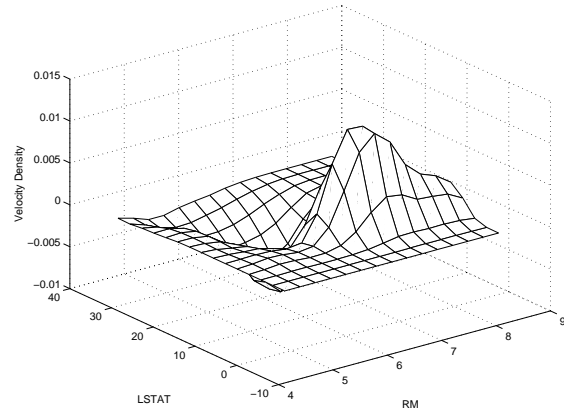


Figure 9: Velocity Density for a highly evolving 2-dimensional Projection (Housing Data Set)

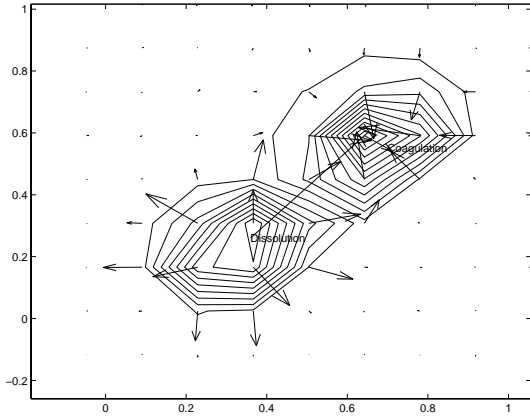


Figure 7: The Spatial Velocity Profile (S-Stream)

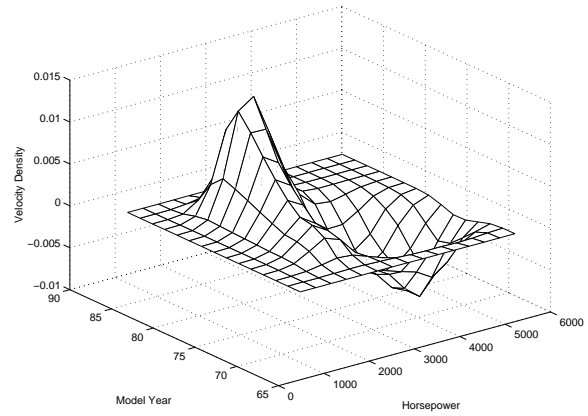


Figure 10: Velocity Density for a highly evolving 2-dimensional Projection (Auto-mpg Data Set)

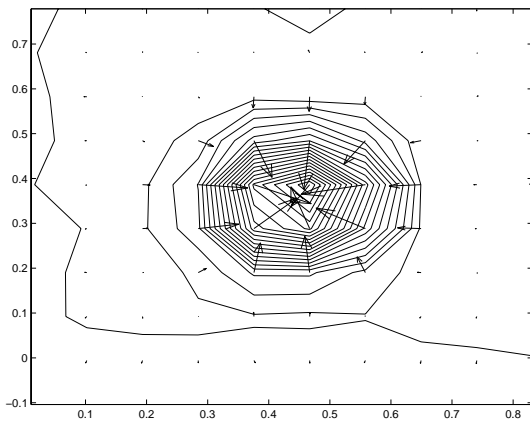


Figure 8: The Spatial Velocity Profile (C-Stream)

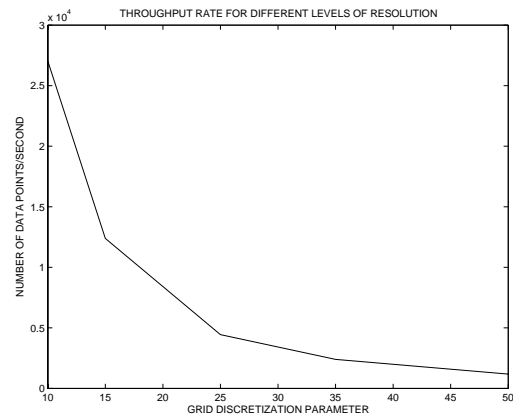


Figure 11: Computational Scalability with Resolution Level

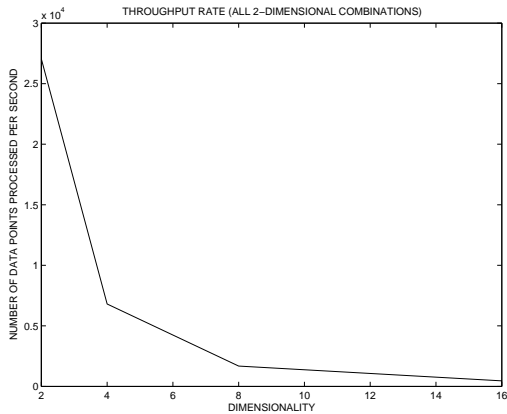


Figure 12: Computational Scalability with number of dimensions

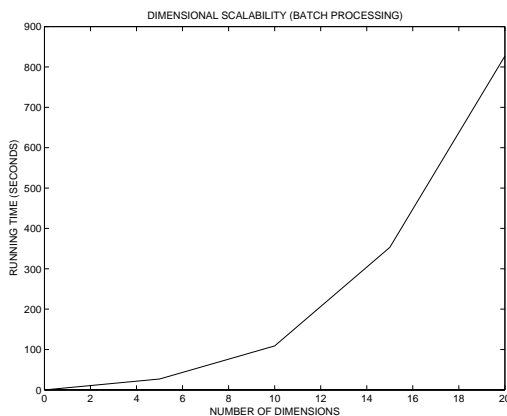


Figure 13: Computational Scalability of Batch Processing Algorithm (Varying Dimensionality)

use of such profiles in order to identify interesting phenomena such as coagulations, dissolutions and shifts. **(3)** Interesting combinations of dimensions in which the greatest amount of data evolution occurs. **(4)** Computational Results illustrating the efficiency of the method.

In our description in the earlier sections we set the values of *min-coag*, *min-dissol* and *min-vel* to be user-defined in the interest of greater flexibility. It is also possible to provide some system guidance in picking them effectively. Let $E_{(h_s, h_t)}(t)$ be the evolution coefficient computed using the γ grid coordinates. Then the average velocity density at any grid point in the data set is given by $E_{(h_s, h_t)}(t)/\gamma \cdot h_t$. For the purpose of our empirical tests, we chose this to be the value of *min-coag* and *min-dissol*. The value of *min-vel* was conservatively set at zero.

First, we used some simple cases with synthetic data sets in order to illustrate the intuitive nature of the results. The first data set consisted of a shifting cluster embedded in a uniformly distributed 2-dimensional data stream. The uniform distribution was spread over the unit square. Each dimension of the cluster was distributed around the center of the cluster with a uniform distribution of total range 0.2. This center shifted in a straight line at an angle of 45 degrees to each axis. We refer to this stream as S-Stream. We have illustrated in temporal and spatial velocity profiles in Figures 6 and 7 respectively. As evident from the charts, there was one region of coagulation and one region of dissolution. This also resulted in one valid shift line. It is also clear from Figure 7 that there was a data shift along the true direction of cluster shifting. We also note that while the static part of the data was uniformly distributed, its nature did not affect the profiles substantially. For example, when the uniformly distributed part of the stream was replaced by static clusters, the temporal and spatial profiles continued to be similar. Thus, the approach measures only the changing part of the data, as opposed to the base distribution.

A second stream was generated in which a cluster materialized in the data at a given location. Thus, the center of the cluster was fixed, whereas the probability of a point belonging to the cluster increased with time. For every 1000 data points, the probability of a point belonging to this cluster increased by 0.005. As in the previous case, the base data was uniformly distributed in the unit square. The cluster had a range of 0.2. We refer to this stream by C-Stream. The spatial profile for this case is illustrated in Figure 8. It is clear that in this case, there was a single area of coagulation corresponding to the cluster of increasing size. A similar result can also be demonstrated for a stream in which the cluster size reduces. The only difference is that in this case, it creates a region of dissolution as opposed to a coagulation.

We also tested the evolution detection system with a number of real data sets. An important feature provided by this framework is to find those combinations of dimensions which show the greatest amount of global evolution. How do we empirically show that the combinations of dimensions found by the algorithm are interesting and meaningful? In order to do so, we generated a data stream from a (real) static data set by using a feature ordering technique. We removed one of the attributes from the data, and generated the remaining data set in the order of the value of this particular attribute. Since the attributes in most real data sets are correlated, the observation of the most highly evolving projections provides an insight into those combinations of attributes which influ-

ence the stripped attribute the most. We emphasize that the only purpose of generating the data in this biased way is to establish the meaningfulness of the projections which show the greatest amount of evolution.

We tested the evolution algorithm using the housing data set of the UCI machine learning repository.⁴ The data was sorted in the order of increasing housing price in order to test which projections of the data which had the greatest amount of evolution. On examining the projections which showed the greatest amount of change, we found some interesting trends. For example, the 2-dimensional projection with the largest evolution coefficient was (RM, LSTAT). Here “RM” stands for the average number of rooms per dwelling, and “LSTAT” stands for the percentage lower status of the population. Clearly both of these attributes have a high relationship to the price of the houses in a given locality, as the price of the houses in a given locality increases with the number of rooms per dwelling and it reduces with increasing percentage lower status of the population. The corresponding temporal velocity profile (see Figure 9) clearly shows this trend. To make a reverse argument, if we had an application in which the data stream showed the kind of trend illustrated in Figure 9, then even without having access to the variable on housing price, one could infer that the stream was gradually getting biased towards more and more expensive housing localities. The provision of algorithmic and visual aid in order to make such diagnosis and understanding can be critical for many commercial applications. One of the interesting aspects that we noted in the data stream was that the variable LSTAT tended to be quite important since it occurred in a number of combinations of dimensions showing high evolution rates. This is because LSTAT directly reflected the financial status of the population; a factor which is closely related to the biased way in which the stream was generated. Other interesting combinations of dimensions which showed high evolution rates were (NOX, DIS) and (NOX, RAD). Here “NOX” corresponds to the nitric oxides concentration, “DIS” corresponds to the weighted distance to five Boston employment centers and “RAD” corresponds to the index of accessibility to radial highways. Again, these combinations of dimensions have high influence on the housing price, which was used to bias the data stream.

Another data set on which we tested the algorithm was the Auto-Mpg data set from the UCI machine learning repository. This data set contained a set of records of different cars along with their mileage (mpg) rates. The numeric attributes contained features such as displacement, horsepower, acceleration, weight and model year of a car. We again intentionally biased the data stream using the mpg attribute so that the records were in increasing order of mpg. Then, we tested the evolution of different combinations of the remaining dimensions. We found that the most highly evolving projection was the combination (horsepower, model year). The corresponding temporal velocity profile is illustrated in Figure 10. An interesting trend from the graph is that the later arrivals correspond to lower horsepower but increasing model year. Again, this seems to reveal the fact that the data was biased in increasing order of the mileage/gallon. Thus, in both cases, by mining the most highly evolving projections it was possible to make infer-

⁴The real data sets used in this paper are available at <http://www.cs.uci.edu/~mlearn>

ences about the nature of the bias in the stream generation.

In order to test the computational scalability, we generated a data set in a similar way as S-Stream, except that the data was generated in 20 dimensions. We refer to this new stream as SM-stream. The process of velocity density calculation requires us to maintain the forward and reverse time slice densities. This is a simple additive process and is also the only process which is dependent on the stream arrival rate. Any additional computation in order to calculate the spatial and temporal velocity profiles from these values needs to be done only once for each temporal window used. This is an (asymptotically) negligible overhead for a fast data stream. In order that the technique can work effectively, the amount of time required in order to calculate the contribution for a given data point to the forward and reverse time slice densities cannot be higher than the rate at which the stream arrives. This ensures that a user can avail himself of the online trends in the data as soon as the temporal window has passed. First, we present the computational requirements of a technique in which the user is presented with the non-overlapping velocity density profiles at periodic intervals of $h_t = 60$ seconds.

In order to calculate the viability of the technique, we computed the *throughput rate* which could be supported by the stream. The throughput rate is the number of data points processed per second for a given value of the grid discretization parameter. We assumed that the time window h_t in which the trends were processed is significantly larger than the interarrival rate between two data points. Successive profiles were generated at times $h_t, 2 \cdot h_t \dots k \cdot h_t$. Since h_t is significantly larger than the interarrival rate, the amount of time required by the final postprocessing procedure at time $i \cdot h_t$ in order to compute the profiles from the densities is (asymptotically) small compared to the time required to process the data points. Therefore, we used the throughput rate as the inverse of the time required in order to process each data point for the velocity density calculation. This is sensitive to the level of discretization used. The higher the value of the discretization parameter, the fewer the number of data points per second which can be processed, but the greater the level of accuracy and refinement. The trends for S-Stream are indicated in Figure 11. The algorithm scales quadratically with the grid discretization parameter. Note that because of the simplicity of the techniques used, thousands of data points can be processed per second at modest values (such as 10 or 15) of the grid discretization parameter. When the rate of arrival of a data stream is higher than the rate that can be supported by the computational resources available, one may choose to either sample selectively in order to maintain the density values, or may choose to reduce the level of refinement level at which the velocity densities are maintained.

Note that the results of Figure 11 are for a single projection of the dimensions in the data. In order to provide visual insights into the most highly evolving 2-dimensional projections of a high dimensional application we need to maintain the velocity density profiles of all pairs of 2-dimensional combinations. In Figure 12, we have shown the scalability of the velocity density estimation technique to increasing data dimensionality, when the grid discretization parameter β is fixed at 10. In order to create data streams of different dimensionalities, projections of different dimensionalities were picked from SM-Stream. Since a total of $\binom{d}{2}$ combina-

tions of projections need to be maintained, the running time per data point scales quadratically with data dimensionality. Again thousands of data points per second can be supported for modest applications.

We also tested the batch processing component of the algorithm in order to find how the computational requirements varied with data dimensionality. In this case, a fixed set of 1000 data points from SM-Stream was used for the computation. We have illustrated the corresponding computational results in Figure 13. Note that even though the number of evolving combinations of dimensions increases exponentially with dimensionality, we are only interested in minimal evolving projections. In practice, only a small subset of the relevant combinations of dimensions are explored. As a result, the rate of increase of the computational time with dimensionality is significantly lower. This behavior is reflected in Figure 13.

7. CONCLUSIONS AND SUMMARY

In this paper we discussed novel techniques for diagnosing different kinds of trends in evolving data streams. The velocity density estimation technique is used in order to generate two kinds of profiles: *spatial velocity profiles* and *temporal velocity profiles*, each of which provide a different kind of insight into the nature of the changes in the underlying data characteristics. We proposed techniques for finding useful trends in higher dimensional problems by finding specific combinations of dimensions in which the greatest change has occurred. Finally, the techniques are used in order to provide an effective diagnosis of the regions of coagulations, dissolutions and shifts in the data. With a few exceptions, most of the results of this paper can be computed in a single online scan of the data. The results of this paper have considerable implications for the increasingly important problem of understanding trends in data streams and spatio-temporal data.

8. REFERENCES

- [1] N. Andrienko, G. Andrienko, P. Gatalisky. Towards Exploratory Visualization of Spatio-Temporal Data. *Third AGILE Conference on Geographical Information Science*, pages 137-142, May 25-27, 2000.
- [2] D. Bonachea, K. Fisher, A. Rogers, F. Smith. Hancock: A language for processing very large data. In *USENIX 2nd Conference on Domain-Specific Languages*, pages 163-176, October 1999.
- [3] R. Agrawal, R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. *VLDB Conference Proceedings*, 1994.
- [4] S. Chawathe, H. Garcia-Molina. Meaningful Change Detection in Structured Data. *ACM SIGMOD Conference Proceedings*, 1997.
- [5] D. Cheung, J. Han, V. Ng, C. Y. Wong. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. *IEEE ICDE Conference Proceedings*, 1996.
- [6] C. Cortes, K. Fisher, D. Pregibon, A. Rogers, F. Smith. Hancock: A Language for Extracting Signatures from Data Streams. *ACM KDD Conference Proceedings*, 2000.
- [7] P. Domingos, G. Hulten. Mining High-Speed Data Streams. *ACM KDD Conference Proceedings*, 2000.
- [8] D. Donjerkovic, Y. E. Ioannidis, R. Ramakrishnan. Dynamic Histograms: Capturing Evolving Data Sets. *IEEE ICDE Conference Proceedings*, 2000.
- [9] R. Feldman, Y. Aumann, A. Amir, H. Mannila. Efficient Algorithms for Discovering Frequent Sets in Incremental Databases. *DMKD Workshop Proceedings*, 1997.
- [10] V. Ganti, J. Gehrke, R. Ramakrishnan. Mining Data Streams under Block Evolution. *ACM SIGKDD Explorations*, 3(2), 2002.
- [11] V. Ganti, J. Gehrke, R. Ramakrishnan, W.-Y. Loh. A Framework for Measuring Differences in Data Characteristics. *ACM PODS Conference Proceedings*, 1999.
- [12] W. Labio, H. Garcia-Molina. Efficient Snapshot Differential Algorithms for Data Warehousing. *VLDB Conference Proceedings*, 1996.
- [13] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.
- [14] J. F. Roddick et al. Evolution and Change in Data Management: Issues and Directions. *ACM SIGMOD Record*, 29(1): 21-25, 2000.
- [15] J. F. Roddick, M. Spiliopoulou. A Bibliography of Temporal, Spatial, and Spatio-Temporal Data Mining Research. *ACM SIGKDD Explorations*, 1(1), June 1999.
- [16] T. Sellis. Research Issues in Spatio-temporal Database Systems. *Symposium on Spatial Databases Proceedings*, 1999.
- [17] S. Thomas, S. Bodagala, K. Alsabti, S. Ranka. An Efficient Algorithm for the Incremental Updating of Association Rules in Large Databases. *ACM KDD Conference Proceedings*, 1997.
- [18] B.-K. Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, A. Biliris. Online Data Mining for Co-Evolving Time Sequences. *IEEE ICDE Conference Proceedings*, 2000.