Shape Sensitive Geometric Monitoring

Izchak Sharfman
Technion
Haifa, Isreal

Assaf Schuster
Technion
Haifa, Israel

Daniel Keren
Haifa University
Haifa, Israel

ABSTRACT

A fundamental problem in distributed computation is the distributed evaluation of functions. The goal is to determine the value of a function over a set of distributed inputs, in a communication efficient manner. Specifically, we assume that each node holds a time varying input vector, and we are interested in determining, at any given time, whether the value of an arbitrary function on the average of these vectors crosses a predetermined threshold.

In this paper, we introduce a new method for monitoring distributed data, which we term *shape sensitive geometric monitoring*. It is based on a geometric interpretation of the problem, which enables to define local constraints on the data received at the nodes. It is guaranteed that as long as none of these constraints has been violated, the value of the function does not cross the threshold. We generalize previous work on geometric monitoring, and solve two problems which seriously hampered its performance: as opposed to the constraints used so far, which depend only on the current values of the local input vectors, here we incorporate their temporal behavior into the constraints. Also, the new constraints are tailored to the geometric properties of the specific function which is being monitored, while the previous constraints were generic.

Experimental results on real world data reveal that using the new geometric constraints reduces communication by up to three orders of magnitude in comparison to existing approaches, and considerably narrows the gap between existing results and a newly defined lower bound on the communication complexity.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: Distributed Systems

General Terms

Algorithms, Performance

PODS'08, June 9–12, 2008, Vancouver, BC, Canada.

1. INTRODUCTION

Many emerging applications require processing high-volume streams of data. Examples include network traffic monitoring systems, real-time analysis of financial data [32, 34], distributed intrusion detection systems, and sensor networks [23]. These applications are commonly referred to as *Data Stream Management Systems* (DSMS) [4]. A key difference between DSMS and traditional DBMS is that DSMS are required to process *continuous queries*. DBMS receive queries that are static in nature, i.e. the system receives a query, and returns a response based on the data currently present in the system. DSMS, on the other hand, are designed to handle continuous queries, i.e. the system receives a query and continuously updates the user as new data arrives. This key difference poses new fundamental challenges that are not addressed by traditional DBMS.

Various types of continuous queries have been studied in the past, including continuous versions of selection and join queries [24], various types of aggregation queries [2, 26], and monitoring queries [6]. While most previous work regarding data stream systems considers sequential setups (the data is processed by a single processor), many data stream applications are inherently distributed: examples include sensor networks [23], network monitoring [17], and distributed intrusion detection.

In many cases, the user of a distributed stream system is interested in receiving notifications when global "events of interest" occur. These tasks are referred to as *distributed monitoring tasks*. Consider, for example, a distributed system for detecting denial of service attacks. A node is considered to be under a denial of service attack if more than a certain percentage of the incoming traffic, say 0.1 percent, is directed to that node. The system is comprised of agents installed on the routers controlling the network traffic entering a local network. Each agent monitors the traffic flowing into the network through its host.

It is easy to see that in the example given above, when a certain node is under a distributed denial of service attack, at least one of the agents will detect that the network traffic to that node exceeds 0.1 percent of the incoming traffic through its host. In other words, the global "event of interest" has a local indication. This fact can be utilized to develop efficient monitoring algorithms, as described in [17].

In more complex monitoring tasks, global "events of interest" may be harder to detect by solely examining local data. As an example of a complex monitoring task, consider a distributed search engine. The engine is comprised of a distributed set of mirrors. Each mirror receives a stream of

queries, where each query consists of multiple search terms. We are interested in monitoring the correlation between the appearance of pairs of search terms is a search query. In order to do so, each mirror keeps track of the queries it received in the last, say, 48 hours. We refer to these queries as the sliding windows held by the mirrors. Given two search terms, denoted A and B, let f_A and f_B be the respective global frequency of occurrence[1] of A and B, i.e. the frequency of their occurrence in the union of the sliding windows held by the mirrors. In addition, let f_{AB} be the global frequency of occurrence of both A and B. The correlation between the appearance of A and that of B is measured using the correlation coefficient ρ_{AB}, which is given by:

$$\rho_{AB}(f_A, f_B, f_{AB}) = \frac{f_{AB} - f_A f_B}{\sqrt{(f_A - f_A^2)(f_B - f_B^2)}}$$

The correlation coefficient receives values in the range $[-1..1]$. A negative score indicates that the terms tend to exclude each other, a score of zero indicates that there is no correlation between the appearance of the terms, and a positive score indicates that the terms tend to appear in the same queries. We would like to receive a notification when the correlation coefficient crosses a given positive threshold. It is easy to see that it is impossible to reach a correct decision, given only the local values of the correlation coefficient. This is a characteristic of non-linear functions.

In this paper we consider a set of nodes, each of which holds a time varying data vector. The global "event of interest" is defined by an arbitrary (possibly non-linear) function over the weighted average of the vectors held by the nodes, and we are interested in detecting when the value of this function crosses a predetermined threshold value. We refer to these monitoring tasks as *non-linear monitoring tasks*. Note that the aforementioned problem of determining the correlation coefficient is covered by this definition.

1.1 Distributed Geometric Monitoring

While non-linear monitoring tasks can be performed by a naive algorithm that collects all the data to a central location for analysis, the communication load incurred by such an algorithm may be prohibitively high. Sensor networks are particularly sensitive to a high communication load since communications are the primary factor affecting the power consumption of the network [33]. In addition to communication load concerns, collecting data to a central location may violate privacy requirements in certain applications.

Previous work [30] has proposed algorithms for performing non-linear monitoring tasks that are based on geometric techniques. The idea is to use the geometric properties of the local data vectors to construct a set of local constraints. Each node verifies that its local data vector conforms to a local constraint. These constraints can be verified independently (i.e. each node can verify its local constraint without communicating with other nodes), and if all the constraints are upheld, the threshold function is guaranteed to remain unchanged. The constraints proposed in [30], however, have several drawbacks.

The first deficiency of these constraints is that they are constructed solely according to the current values of the local data vectors. Real-world data usually displays an underlying distribution. We fit a probabilistic model to this distribution, and use it to create local constraints that are optimized to the data received on the nodes, in the sense that the probability of a constraint violation is minimized.

Another disadvantage of geometric constraints proposed in previous work is that they are *generic* in the sense that the same constraints are used regardless of the function at hand. We present a new approach, which allows to construct a range of valid constraints. This poses a new challenge – how to select the constraints that are optimal given a certain function. We address this challenge, and offer a method for choosing the optimal constraints.

1.2 Contributions

The contributions of this work are the following:

1. We fit a distribution to the data received by the nodes, and use this distribution to create geometric constraints that minimize the probability of violation.

2. For every constraint, we define a formal construct called a "safe zone", which allows to measure the constraint's effectiveness in relation to the given function. This measure, in turn, is used to select an effective constraint.

3. We present a theoretic optimal constraint (i.e. when it is violated, any other possible geometric constraint will be violated as well). We use this optimal constraint to evaluate how much further improvement to our constraints is at all possible.

4. We present experimental results on real world data that show that the constraints we propose can lead to a reduction of up to three orders of magnitude in communication in comparison to the existing geometric constraints. Using the new constraints drastically reduced the gap between previous results, and those achieved when using theoretic optimal constraints.

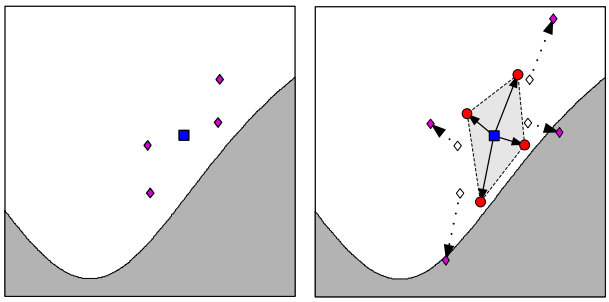
2. RELATED WORK

A well studied problem is the monitoring of frequency counts over a single data stream [2, 3, 8, 26], however these works do not address distributed environments. Other important problems over a single data stream were studied in [7, 9, 16, 18].

Distributed function computation has been addressed by the "Distributed Triggers" framework presented in [21]. Later, this framework has been employed to monitor network wide traffic anomalies [19, 20]. Our work is consistent with the distributed triggers framework in that it employs a set of local constraints for detecting a global event of interest. In contrast to [19, 20], which focus on an anomaly detection problem, our work addresses a wide class of non-linear monitoring problems.

A well studied problem is the computation of a the sum or average of a distributed set of variables. Prominent examples include [17], which studies detecting when such a sum exceeds a given threshold, [10], which proposes observing the distribution of the input data to receive optimal algorithms, and [27] which enables tracking the sum, average, or minimum of a distributed set of variables within a certain predetermined error margin. Our work differs in that we

[1]Frequency of occurrence refers to the number of search queries containing the term divided by the total number of queries.



(a) (b)

Figure 1: Geometric interpretation of a monitoring problem.

monitor the value of an arbitrary function over a vector of distributed variables.

A common approach to distributed stream monitoring is the use of sketches to summarize data while maintaining accuracy guarantees. This approach has been employed to detect "heavy hitters" [25], compute quantiles [12], count distinct elements [14], and compute join aggregates [11].

Other distributed computation problems studied in previous work include top-k problems [5], set-expression cardinality estimation [15], clustering [13], and distributed verification of logical expressions [1].

Recently, the geometric monitoring scheme was proposed in [30]. In contrast to the methods proposed in [30], which are oblivious to the nature of the data on the streams and to the monitored function at hand, the methods presented here leverage this information, yielding very significant reductions in communications.

3. THE MONITORING FRAMEWORK

We now present a general framework for performing non-linear monitoring tasks. Given a monitoring task, we refer to the data held by each node as the *local data vector*, and to their weighted average as the *global data vector*. Denote the dimension of the vectors by d. We refer to the function evaluated at the global data vector as the *monitored function*. The combination of the monitored function and the threshold value is viewed as inducing a coloring over the d-dimensional domain: vectors for which the value of the monitored function is above the threshold are colored white, and vectors for which the value is below it are colored gray. Given the geometric interpretation, the goal of the monitoring task can be viewed as determining the color of the global data vector at all times.

Upon initialization of the monitoring task, and from time to time, as dictated by the monitoring algorithm, all the local data vectors are collected by a certain node designated as the *coordinator*. The coordinator calculates the weighted average of the local data vectors (i.e. determines the global data vector), and sends this value to the nodes. We refer to this vector as the *estimate vector*, and denote it by $\vec{e}$. The process of collecting the local data vectors and calculating the estimate vector is referred to as a *synchronization process*.

As data arrives on the streams maintained by the nodes, each node keeps track of the difference between the current value of its data vector and its value at the time of the last synchronization process. We refer to this difference as the *delta vector*. We denote the sum of the estimate vector and the delta vector as the *drift vector* . The drift vector held by the i^{th} node is denoted by $\vec{v}_i$. It is easy to verify that the global data vector is a convex combination of the drift vectors, hence it belongs to their convex hull.

Figure 1(a) depicts the coloring induced by the combination of the monitored function and the threshold value, the initial data vectors (purple diamonds) and the estimate vector (blue square). As data arrives at the nodes, the data vectors change. The new location of the data vectors (purple diamonds), as well as their initial location (white diamonds) are depicted in Figure 1(b). In addition, the corresponding drift vectors (red circles) are depicted, and their convex hull is highlighted in gray. Note that the difference between the current and initial values of the a data vector is equal to the difference between the corresponding drift vector and the estimate vector.

Our goal is to construct a set of local constraints on the values of the drift vectors, such that each node can verify its local constraint independently (i.e. without communicating with other nodes), and if all the constraints are upheld, the convex hull of the drift vectors is guaranteed to be monochromatic (i.e. all the vectors in it are of the same color). As long as the convex hull of the drift vectors is monochromatic, each node knows that the color of the global data vector is equal to the color of its drift vector, i.e. it knows the value of the function is above or below the threshold.

The local constraints are regions in $\mathbb{R}^d$ that each node determines according to the estimate vector and its drift vector. If the region determined by a node is monochromatic, the constraint is upheld, otherwise it is violated. We call these regions *bounding regions*, since we require that their union covers the convex hull of the drift vectors.

We propose the following method for constructing bounding regions: the nodes agree on a $d \times d$ symmetric positive definite matrix A, which is called the parameters matrix. In addition, the nodes agree on a common vector denoted by $\vec{r}$, which is called the reference vector. The parameters matrix and reference vector are determined during the synchronization process. Given a node's drift vector $\vec{v}_i$, the reference vector $\vec{r}$, and the parameters matrix A, the bounding region constructed by the node is the ellipsoid E_A, which is defined as follows:

$$E_A(\vec{r},\vec{v}_i) = \left\{ \vec{z} \middle| \left(\vec{z} - \tfrac{\vec{r}+\vec{v}_i}{2}\right)^T A \left(\vec{z} - \tfrac{\vec{r}+\vec{v}_i}{2}\right) \leq \left(\tfrac{\vec{r}-\vec{v}_i}{2}\right)^T A \left(\tfrac{\vec{r}-\vec{v}_i}{2}\right) \right\} \quad (1)$$

Note that by setting the parameters matrix to the unit matrix, the bounding region maintained by each node is a sphere centered at the midpoint between the reference vector and the node's drift vector, whose radius is half the distance between the reference vector and drift vector. Given a reference vector and a drift vector, we denote the sphere created by using a unit matrix as the parameters matrix by $B(\vec{r},\vec{v}_i)$:

$$E_I(\vec{r},\vec{v}_i) = B(\vec{r},\vec{v}_i) = \left\{ \vec{z} \middle| \left\| \vec{z} - \frac{\vec{r}+\vec{v}_i}{2} \right\|_2 \leq \left\| \frac{\vec{r}-\vec{v}_i}{2} \right\|_2 \right\}$$

Using a unit matrix as the parameters matrix and the

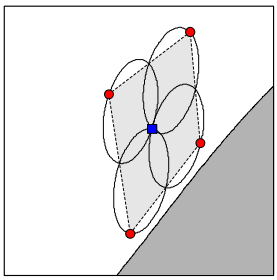


Figure 2: The use of ellipsoidal constraints.

estimate vector as the reference vector yields the spherical bounds presented in [30].

When the parameters matrix is not a unit matrix, the bounding region held by a node is an ellipsoid centered at the midpoint between the reference vector and the nodes drift vector. Note that the ellipsoidal bounding regions held by the nodes have the same orientation (their respective axes are parallel), and the same proportions. Figure 2 illustrates the use of local constraints to determine the value of the threshold function. The coloring induced by the combination of the monitored function and the threshold value is depicted. The convex hull of the drift vectors (red circles) is highlighted, and the ellipsoids constructed by the various node are illustrated. As illustrated in the figure, all the ellipsoids are monochromatic, guaranteeing that the convex hull is monochromatic as well.

In summary, the monitoring algorithm proceeds as follows: upon initialization, a synchronization process is performed, after which each node holds an estimate vector, a reference vector, and a parameters matrix. Note that after the synchronization process, all the drift vectors are equal to the estimate vector. If the estimate vector is used as the reference vector, the ellipsoids constructed by the nodes are a single vector (and hence monochromatic), therefore after the synchronization process, all the constraints are upheld. As more data arrives on the streams, each node verifies that its ellipsoid is monochromatic. If one of the ellipsoids is not monochromatic, a synchronization process is performed.

We show that choosing the optimal parameters matrix enables us to customize the constraints to the properties of the data received on the streams. In addition, we show that by carefully selecting the reference vector, we can customize the constraints to the monitored function at hand.

In the following section we discuss the use of ellipsoidal constraints. We build a probabilistic model of the data, and use it to determine the parameters matrix. In addition, we show that the ellipsoidal constraints are valid, i.e. the union of the ellipsoids constructed by the nodes is guaranteed to cover the convex hull of the drift vectors.

4. CONSTRUCTING DATA SENSITIVE CONSTRAINTS

The goal of this section is to construct bounding regions that cover the convex hull of the drift vector as tightly as possible. We thus decrease the number of "false positives" generated by the constraints. A geometric constraint causes a "false positive" if the bounding region it defines is not monochromatic, while the convex hull of the drift vectors is monochromatic.

We use previous data received on the streams to construct a probabilistic model of future values, and then use this model to construct optimal tight bounding regions.

4.1 Data Modeling

In order to model the data received on the streams, we view it as if it were generated by a probabilistic data source. Consider, for example, the search term correlation example presented in Section 1. Let us assume that the appearance of each search term in a query is determined by a stationary random variable that receives 1 if the term appears in the query, and 0 otherwise. We assume that the terms used in a certain query are drawn independently of the terms used in previous queries (note that we do not assume that the values drawn for the various terms in a certain query are independent). Consequently, given two search terms, A and B, the global data vector (f_A, f_B, f_{AB}) can be viewed as an average of i.i.d random vectors.

Under these assumptions, according to the Central Limit Theorem, the distribution of the global data vector converges to a multi-variate Gaussian distribution (as the number of search queries held in the sliding windows increases). Recall the multi-variate Gaussian distribution:

$$G_{\vec{\mu},\mathbf{\Sigma}}(\vec{v}) = \frac{1}{(2\pi)^{d/2}\det(\mathbf{\Sigma})^{1/2}} \exp(-\frac{1}{2}(\vec{v}-\vec{\mu})^T\mathbf{\Sigma}^{-1}(\vec{v}-\vec{\mu}))$$

Were $\vec{\mu}$ is the expected value of the global data vector, and $\mathbf{\Sigma}$ is its covariance matrix. Given a set of previous values of the drift vectors, we can build an empirical probability distribution of these values by calculating their mean and covariance matrix, and using them as the parameters of a multi-variate Gaussian distribution. Since we assumed that the data is stationary, we can use this distribution to predict future values of the drift vectors.

Note that a Gaussian distribution of drift vectors is not particular to the search term correlation example. The Gaussian distribution is commonly used to characterize the behavior of natural phenomena, and is particularly suitable whenever the drift vector can be modeled as the sum or average of independently drawn random vectors. As a result, we expect that the methods presented below will be applicable to a large family of practical problems.

4.2 Using the Model to Construct Tight Bounding Regions

Intuitively, it would be simpler to construct tight bounding regions on the drift vectors if the data were isotropic, i.e. transformed so that it is distributed evenly in all directions. Formally, this means that the variances of the transformed data along the various axes is identical, and that the covariance among any pair of axes is zero. We use the model we built to determine a transformation that normalizes the data so that it is isotropic. Once this transformation is determined, it can be applied independently on the data received at each node. We show that by first applying this transformation on the drift vectors, and then using spherical bounding regions as described in Section 3, we receive ellipsoidal bounding regions. In addition, we prove the validity of the ellipsoidal constraints (i.e. the ellipsoidal bounding regions are guaranteed to cover the convex hull of the drift vectors), and show that by applying the normalizing transformation, we receive the tightest possible ellipsoidal bounding regions.

We start by describing the normalization process. Given $G_{\vec{\mu},\mathbf{\Sigma}}(\vec{v})$, a multi-variate Gaussian distribution of the drift vectors, we wish to transform the data so that the variance of the transformed data is identical along the various axes, and the covariance among any pair of of axes is zero. In order to do so, observe that the covariance matrix $\mathbf{\Sigma}$ is symmetric and positive definite, and can therefore be decomposed as follows:

$$\mathbf{\Sigma} = P_{\mathbf{\Sigma}} D_{\mathbf{\Sigma}} P_{\mathbf{\Sigma}}^{-1}$$

Where P_Σ is a matrix whose columns are the normalized eigenvectors of Σ, and D_Σ is a diagonal matrix with the respective eigenvalues on its diagonal. Since Σ is symmetric, its eigenvectors are orthogonal, and therefore P_Σ is orthonormal. Since Σ is positive definite, its eigenvalues are positive. Let D'_Σ be the square root of D_Σ (i.e. a diagonal matrix with the square root of the eigenvalues of Σ on its diagonal). Since $P_{\mathbf{\Sigma}}$ is orthonormal, it can be viewed as a rotation transformation, and $P_{\mathbf{\Sigma}}^{-1}$ can be viewed as the inverse rotation transformation. Since D'_Σ is a diagonal matrix, it can be viewed as a scaling transformation. Let us define the linear transformation $T_\Sigma = {D'}_\Sigma^{-1} P_\Sigma^{-1}$. Note that T_Σ is a concatenation of a rotation transformation and a scaling transformation. Given a drift vector $\vec{v}$, drawn from the distribution $G_{\vec{\mu},\mathbf{\Sigma}}(\vec{v})$, let $\vec{v'}$ be the image of $\vec{v}$ under the transformation T_Σ, i.e. $\vec{v'} = T_\Sigma \vec{v}$. It is easy to show that the distribution of the transformed vectors is $G'_{\vec{\mu'},I}(\vec{v'})$:

$$G'_{\vec{\mu'},I}(\vec{v'}) = \frac{1}{(2\pi)^{d/2}} \exp(-\frac{1}{2}(\vec{v'} - \vec{\mu'})^T(\vec{v'} - \vec{\mu'}))$$

where $\vec{\mu'} = T_\Sigma \vec{\mu}$. Note that the distribution $G'_{\vec{\mu'},I}(\vec{v'})$ is isotropic, i.e. the variance of the data is equal in all directions.

The ellipsoidal bounding regions are constructed as follows: given an estimate vector $\vec{e}$ and a drift vector $\vec{v}_i$, we determine $\vec{e'}$ and $\vec{v'}_i$, their image under the transformation T_Σ. Next we construct a sphere centered at the midpoint between $\vec{e'}$ and $\vec{v'}_i$, with a radius of half the distance between $\vec{e'}$ and $\vec{v'}_i$ (recall that this sphere is denoted by $B(\vec{e'}, \vec{v'}_i)$). Observe that the image of the sphere $B(\vec{e'}, \vec{v'}_i)$ under the transformation T_Σ^{-1} is the ellipsoid $E_{\Sigma^{-1}}(\vec{e}, \vec{v}_i)$ (see Equation 1). According to a theorem from [30] (which is quoted below), is follows that given a set of drift vectors $\vec{v}_1,...,\vec{v}_n$, the union of the spheres $B(\vec{e'}, \vec{v'}_1),...,B(\vec{e'}, \vec{v'}_n)$ bounds the convex hull of the vectors $\vec{v'}_1,...,\vec{v'}_n$:

THEOREM 1. *Let $\vec{x}, \vec{y}_1, \vec{y}_2, ..., \vec{y}_n \in \mathbb{R}^d$ be a set of vectors in $\mathbb{R}^d$. Let $Conv(\vec{x}, \vec{y}_1, \vec{y}_2, ..., \vec{y}_n)$ be the convex hull of $\vec{x}, \vec{y}_1, \vec{y}_2, ..., \vec{y}_n$. Let $B(\vec{x}, \vec{y}_i)$ be a sphere centered at $\frac{\vec{x}+\vec{y}_i}{2}$ and with a radius of $\left\|\frac{\vec{x}-\vec{y}_i}{2}\right\|_2$ i.e., $B(\vec{x}, \vec{y}_i) = \left\{\vec{z} \middle| \left\|\vec{z} - \frac{\vec{x}+\vec{y}_i}{2}\right\|_2 \leq \left\|\frac{\vec{x}-\vec{y}_i}{2}\right\|_2\right\}$. Then $Conv(\vec{x}, \vec{y}_1, \vec{y}_2, ..., \vec{y}_n) \subset \bigcup_{i=1}^{n} B(\vec{x}, \vec{y}_i)$.*

Since linear transformations preserve convexity (i.e. the image of the convex hull of a set of vectors is the convex hull of the images of these vectors), it follows that the convex hull of the original drift vectors is covered by the ellipsoids $E_{\Sigma^{-1}}(\vec{e}, \vec{v}_1),...,E_{\Sigma^{-1}}(\vec{e}, \vec{v}_n)$. Figure 3 illustrates the process of bounding the convex hull of a set of vectors using ellipsoids. This process can be thought of as first applying a rotation transformation (1) and then a scaling transformation (2) on the vectors. Next, the convex hull is bound using spheres (3), and finally, the inverse transformations are applied (4,5).

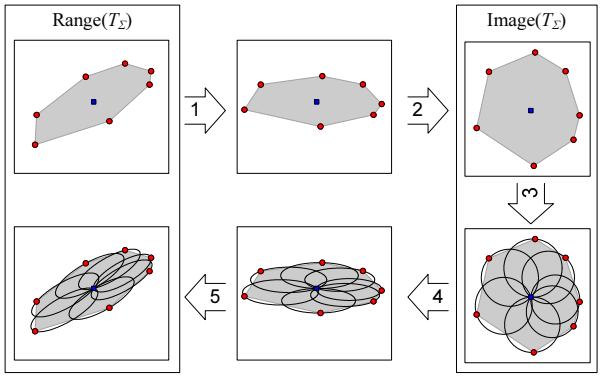


Figure 3: Bounding a convex hull with ellipsoids.

During the synchronization process, in addition to its drift vector, each node sends the coordinator the covariance matrix and mean vector of the data values contained in its sliding window (note that if the dimension d is large, bandwidth can be saved by compactly representing the covariance matrix using the leading terms of its spectral decomposition). The coordinator uses this data to calculate the covariance matrix representing the global data set, sends its inverse to the nodes as the parameters matrix, and sends the estimate vector as the reference vector. Experimental results show that using these ellipsoidal bounding regions can reduce communication by over an order of magnitude in comparison to spherical constraints.

In the following section we formally show that using the inverse of the covariance matrix produces optimal ellipsoidal bounding regions.

4.3 Optimality of the Ellipsoidal Bounds

In the previous section we proposed ellipsoidal bounding regions, using the inverse of the covariance matrix of the data as the parameters matrix. Intuitively, this transformation should be optimal, since using these ellipsoids is equivalent to bounding transformed, isotropic data with spheres. In other words, if we use a different parameters matrix, we should receive ellipsoids whose expected volume is greater. Theorem 2 formally confirms this intuition. For technical reasons, we assume that the parameters matrix is normalized so that its determinant equals 1. We can do this without loss of generality, since multiplying the parameters matrix by a positive constant produces the same ellipsoid.

THEOREM 2. *Let A be a $d \times d$ positive definite matrix such that $\det(A) = 1$. Let $\mathbf{\Sigma}$ be the covariance matrix of a d-dimensional Gaussian distribution $G(\vec{0}, \mathbf{\Sigma})$ centered at the origin. Let $\vec{x}$ be a random vector drawn according to G. Let $V(\vec{x})$ be the volume of the ellipsoid $E_A(\vec{0}, \vec{x})$. Then the expected value of $V(\vec{x})$ is minimized by $A = \frac{\mathbf{\Sigma}^{-1}}{\det(\mathbf{\Sigma}^{-1})^{1/d}}$.*

PROOF. Omitted due to space considerations. □

As long as the data received on the streams has a stationary distribution, choosing $\mathbf{\Sigma}^{-1}$ as the ellipsoid matrix is optimal

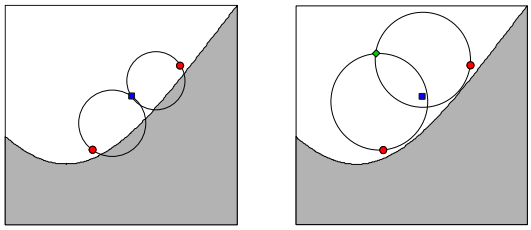


Figure 4: The effect of the reference vector on the bounding regions.

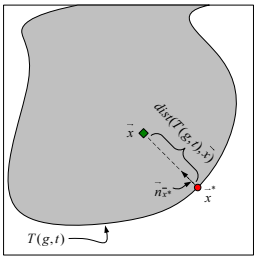


Figure 5: Illustration of a threshold surface, a reference point, and the normal.

in the sense that Theorem 2 guarantees that this parameters matrix minimizes the mean volume of the bounding ellipsoids.

5. STRATEGIES FOR SELECTING A REFERENCE VECTOR

Up to this point, we focused on creating constraints that minimize the volume of the bounding region maintained by each node. While this approach is very effective, it only takes into account the convex hull that is needs to be bounded, and does not consider the monitored function. In fact large, but carefully constructed bounding regions, can be more effective than smaller regions in bounding the convex hull of the drift vectors. The size of the bounding regions is affected by the choice of the reference vector used for constructing them. Up to this point, we used the estimate vector as the reference vector. However, as the following example demonstrates, the estimate vector is not necessarily the best reference vector. Figure 4 depicts the coloring induced by the function $f(x,y) = 2x^2/(x^2+1) - y$, and a threshold value of 0. In addition, it depicts two drift vectors ((0.45,0.39) and (0.105,0.055)) and an estimate vector (0.26,0.268). The figure depicts two choices of reference vectors, in Figure 4(a) the estimate vector is used as the reference vector, while in Figure 4(b), a vector that is more "distant" from the threshold surface is used (the threshold surface is the set of vectors for which the value of the monitored function equals the threshold value). As illustrated by the figure, despite being larger, the spheres created with the "distant" reference vector are monochromatic, while the smaller spheres created with the estimate vector are not.

In this section we will expand upon the intuitive concepts described above. We begin by describing some notations, then define a formal construct called a safe zone. Next, we show how safe zones can be used to evaluate the merits of a given reference vector, and finally, describe a method for selecting good reference vectors.

5.1 Notations

Following are some notations used throughout this section. As mentioned above, a monitored function g and the threshold value t define the threshold surface $T(g,t)$, i.e. the set of vectors for which the value of the monitored function is equal to the threshold value:

$$T(g,t) = \{\vec{x} \,|g(\vec{x}) = t\}$$

We assume that the monitored function g is continuous and differentiable in all $\mathbb{R}^d$.

The distance of a vector $\vec{x}$ to the threshold surface is defined as the minimum over the distances of $\vec{x}$ to all the vectors on the threshold surface, and is denoted by $\text{dist}(T(g,t),\vec{x})$:

$$\text{dist}(T(g,t),\vec{x}) = \min\left(\|\vec{z}-\vec{x}\|_2 \,|\vec{z} \in T(g,t)\right) \qquad (2)$$

Note that a sphere is monochromatic if and only if the distance of its center to the threshold surface is greater than its radius. The vector on the threshold surface that is closest to a given vector $\vec{x}$ is denoted by $\vec{x}^*$. Note that there may be more than one vector on the threshold surface that minimize the distance to $\vec{x}$. In this case $\vec{x}^*$ is arbitrarily selected among these vectors. Denote the normal to the threshold surface at $\vec{x}^*$ by $\vec{n}_{\vec{x}^*}$. These constructs are illustrated in Figure 5.

Given a monitored function g and a threshold value t, define a function $\text{col}_{g,t}(\vec{x})$ as follows:

$$\text{col}_{g,t}(\vec{x}) = \begin{cases} 1 & \text{if } g(\vec{x}) > t \\ 0 & \text{if } g(\vec{x}) = t \\ -1 & \text{if } g(\vec{x}) < t \end{cases}$$

Two vectors $\vec{x}$ and $\vec{y}$ have the same color if $\text{col}_{g,t}(\vec{x}) \cdot \text{col}_{g,t}(\vec{y}) = 1$.

Given a reference vector $\vec{r}$, let $S_{g,t}(\vec{r})$ be the set of all the vectors that create a monochromatic sphere with $\vec{r}$:

$$S_{g,t}(\vec{r}) = \left\{\vec{z} \left| \left\| \frac{\vec{z}-\vec{r}}{2} \right\|_2 < \text{dist}\left(T(g,t), \frac{\vec{z}+\vec{r}}{2}\right)\right.\right\} \qquad (3)$$

We call this set the *safe zone* induced by g, t, and $\vec{r}$.

5.2 Using Safe Zones to Evaluate Reference Vectors

In this section, we assume that spherical bounding regions are used to bound the convex hull of the drift vectors. In Section 5.3, we will discuss how the techniques developed in this section can be applied to ellipsoidal bounding regions as well. Figure 6 illustrates the safe zone defined by the estimate vector and threshold surface that were depicted in Figure 4 (outlined in blue), as well as the safe zone defined by the "distant " reference vector (outlined in green). It is evident from the illustration that the safe zone induced by the "distant" reference vector includes the safe zone induced by the estimate vector. In other words, any drift vector that creates a monochromatic sphere with the estimate vector creates a monochromatic sphere with the "distant" reference vector as well, but there is also a large set of drift vectors that create a monochromatic sphere with the "distant" reference vector but not with the estimate vector. In this regard,

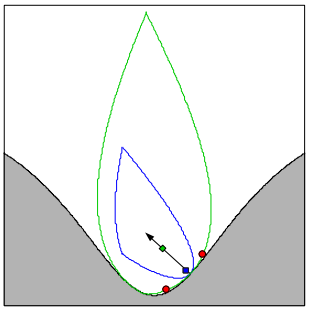


Figure 6: The estimate vector (blue) and the "distant" vector (green) that were depicted in Figure 4.

the "distant" reference vector is a better choice of reference vector than the estimate vector.

Safe zones enable us to formally evaluate the merits of a certain choice of reference vector in relation to a given function. For example, as described above, if the safe zone induced by a given reference vector contains the safe zone induced by the estimate vector, then the former is obviously a better reference vector then the estimate vector. However, using the full containment of safe zones as a criteria for evaluating reference vectors can be restrictive. Intuitively, we would like to find a reference vector so that the safe zone it induces is large at the "vicinity" of the estimate vector. We are less concerned about whether or not the safe zone induced by a reference vector includes vectors that are far from the estimate vector. This is because after a synchronization process, the drift vectors are equal to the estimate vector. As data arrives at the nodes, the drift vectors tend to concentrate around the estimate vector (this is also supported by the probabilistic model of the data). In order to capture this intuition we define a property we call *local containment of safe zones*. The idea is that given two reference vectors, $\vec{r}_1$ and $\vec{r}_2$, then for $\vec{r}_2$ to be considered better than $\vec{r}_1$, it is sufficient for $S_{g,t}(\vec{r}_2)$ to contain $S_{g,t}(\vec{r}_1)$ in the "vicinity" of the estimate vector. We formally define a *vicinity* of the estimate vector as a connected set[2] of vectors that contains the estimate vector. In addition, we say that a vicinity of the estimate vector is *sufficiently large* if it also contains $\vec{e^*}$, the point on the threshold surface that is the closest to the estimate vector. A formal definition of the local containment property follows.

DEFINITION 1. *Let g be a monitored function, t a threshold value, and $\vec{e}$ an estimate vector. Let L be a vicinity of the estimate vector, i.e. a connected set that includes the estimate vector. Let $\vec{r}_1$ and $\vec{r}_2$ be two reference vectors. We say that the safe zone induced by $\vec{r}_1$ is locally contained in the safe zone induced by $\vec{r}_2$ if $[S_{g,t}(\vec{r}_1) \cap L] \subset [S_{g,t}(\vec{r}_2) \cap L]$.*

We proceed to construct a sufficiently large vicinity of the estimate vector and a reference vector, such that the safe zone induced by it locally contains the safe zone induced by the estimate vector. To achieve this, we examine the set of vectors R that satisfy the following conditions:

[2]The term "connected set" refers to a path-wise connected set, i.e. given two vectors $\vec{x}$ and $\vec{y}$ that belong to the set, there exists a continuous function f, that maps the section [0,1] to members of the set such that $f(0) = \vec{x}$, $f(1) = \vec{y}$,

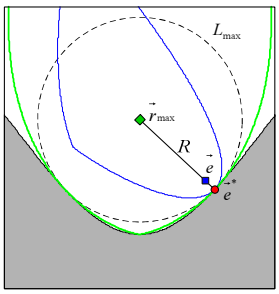


Figure 7: A threshold surface, an estimate vector, the set R and the vicinity L_{max}.

1. For each vector $\vec{r} \in R$, $\vec{e^*}$ is the vector on the threshold surface that is the closest to $\vec{r}$.
2. For each vector $\vec{r} \in R$, $\text{col}_{g,t}(\vec{r}) = \text{col}_{g,t}(\vec{e})$.

Let $\vec{r}_{max}$ be the vector in R that is the most distant from $\vec{e^*}$. We refer to $\vec{r}_{max}$ as the *internal vector* (since, intuitively speaking, it is found by moving from $\vec{e^*}$ into the region – white or gray – that contains $\vec{e}$). Let L_{max} be a sphere centered at $\vec{r}_{max}$, and whose radius is the distance between $\vec{r}_{max}$ and $\vec{e^*}$ (see Figure 7). It is easy to see that L_{max} includes both $\vec{e}$ and $\vec{e^*}$, and is therefore a sufficiently large vicinity of the estimate vector. We propose using $\vec{r}_{max}$ as the reference vector. This has two advantages. The first is that the safe zone induced by $\vec{r}_{max}$ locally contains the safe zone induced by the estimate vector (with respect to the vicinity L_{max}). The second is that given $\vec{e}$, $\vec{r}_{max}$ can be efficiently determined. We start by showing that the safe zone induced by $\vec{r}_{max}$ locally contains the safe zone induced by the estimate vector. This proceeds as follows: in Lemma 1 we show that the safe zone induced by $\vec{r}_{max}$ contains the entire vicinity. As a result, it locally contains the safe zone induced by the estimate vector.

LEMMA 1. *Let g be a monitored function, t be a threshold value, and $\vec{e}$ be an estimate vector. Let $\vec{r}_{max}$ and L_{max} be a reference vector and a sufficiently large vicinity, as described above. Then $L_{max} \subset S_{g,t}(\vec{r}_{max})$.*

PROOF. Omitted due to space considerations. □

Next, we show that $\vec{r}_{max}$ can be efficiently determined. In order to do so, we observe that the vectors in R are located along the normal to the threshold surface at $\vec{e^*}$. This follows immediately from Lemma 2 below. The significance of this observation is that regardless of the dimensionality of the domain of the function g, the vectors in R are a subset of a one-dimensional subspace.

LEMMA 2. *Let g be a monitored function, t a threshold value, $\vec{x}$ an arbitrary vector, and $\vec{x^*}$ the vector on the threshold surface that is the closest to $\vec{x}$. Then $\vec{x}$ lies along the ray starting at $\vec{x^*}$, and whose direction is defined by $\vec{n}_{\vec{x^*}}$, the normal to the threshold surface at $\vec{x^*}$. In other words, there is a real value α, such that $\vec{x} = \vec{x^*} + \alpha\vec{n}_{\vec{x^*}}$.*

PROOF. Ommited due to space considerations. □

The fact that the set R is a subset of a one-dimensional subspace enables employing the following strategy for selecting

a reference vector: during the synchronization process, after determining the estimate vector $\vec{e}$, the coordinator calculates $\vec{e^*}$ (this can be done by employing the optimization techniques described in [28]). Once $\vec{e^*}$ has been calculated, the coordinator sets the reference vector to be equal to the estimate vector, and iteratively examines new reference vectors by doubling the distance of the previous reference vector from $\vec{e^*}$, i.e. the reference vector $\vec{r_i}$ examined in the i^{th} iteration is $\vec{e^*} + 2^i(\vec{e} - \vec{e^*})$. In each iteration we calculate the vector on the threshold surface that is closest to $\vec{r_i}$. If this vector is $\vec{e^*}$, we proceed to the next iteration, and, continuing in this fashion, find $\vec{r}_{max}$ using a binary search. The synchronization process is concluded by sending the nodes $\vec{r}_{max}$ as the reference vector.

Experimental results show that using spherical bounding regions while using the internal vector as the reference vector typically reduces communication by over an order of magnitude in comparison to the spherical constraints used in [30].

5.3 Selecting Reference Vectors while Employing Ellipsoidal Bounding Regions

In Section 4 we leveraged a probabilistic model of the data to reduce the communication load incurred by the monitoring algorithm by employing ellipsoidal constraints. In this section we reduced the communication by selecting a better reference vector, but we assumed that spherical constraints are used. We now describe an algorithm that combines both methods.

Recall that bounding the convex hull of the drift vectors using ellipsoids is equivalent to bounding the convex hull of drift vectors with spheres after applying a linear transformation to the drift vectors. In order for the transformed monitoring problem to remain consistent with the original monitoring problem, we apply the transformation to the monitoring function as well, i.e. if $g(\vec{v})$ is the monitored function, and T_Σ is the transformation applied to the drift vectors, then the transformed monitored function is $g(T_\Sigma^{-1}\vec{v})$.

We can select a better reference vector while leveraging the probabilistic model of the data by selecting the reference vector in the transformed monitoring problem. Note that, as opposed to the case of the isotropic distribution assumed in Section 5.2, here the optimal direction by which to move away from the boundary is not necessarily orthogonal to it. Experimental results show that combining ellipsoidal bounds with reference vector selection yields a reduction in communication that is far greater than employing each of these methods separately. Typically, the reduction in communication achieved by combining both approaches reaches two orders of magnitude in comparison to the spherical constraints presented in [30], and in certain cases exceeds three orders of magnitude.

6. THEORETIC OPTIMAL GEOMETRIC CONSTRAINT

Experimental results show that employing the new geometric constraints presented above dramatically reduce the communication cost incurred by monitoring algorithms. The question arises, how much further improvement to the constraints is at all possible. The geometric monitoring scheme is based on distributively constructing bounding regions, such that the union of these regions covers the convex hull of the drift vectors. Since all the nodes know about the global vector is that it is contained in this convex hull, the most we can expect from any set of geometric constraints is for a constraint to be violated only if the convex hull is is not monochromatic, i.e. a constraint with no "false positives". We refer to such constraints as *optimal constraints*.

We can simulate optimal constraints by collecting all the drift vectors every time new data is received by one of the nodes, and checking whether the convex hull of these vectors is monochromatic. Such an approach is not feasible in practice, since checking the constraints requires the nodes to communicate, but simulating these constraints gives us an indication of how much additional improvement can be expected.

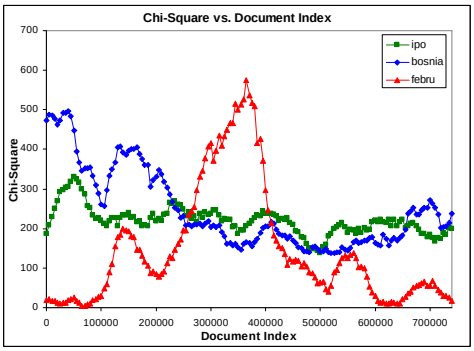


Figure 8: Chi-square score for the features "bosnia", "ipo", and "febru" as it evolves over the streams.

7. EXPERIMENTAL RESULTS

We performed several experiments using various geometric constraints. The constraints were tested on a distributed feature selection problem. In this setup news stories, which are referred to as documents, are received at a set of distributed nodes. Each document is categorized as belonging to several categories (Sports, News, etc.). Our goal is to select the most relevant words, or features, for classifying the documents according to a certain label (e.g. News). This task, which is of great importance in data mining and machine learning, is known as feature selection. In order to decide, at any given time, whether or not to select a certain feature, each node maintains a sliding window containing the last k documents it received. The relationship between the appearance of the feature and the category label is captured using a contingency table, and the relevance of the feature is scored by evaluating the chi-squared measure on the sum of the contingency tables held by the nodes. The feature is selected if its chi-square score exceeds a predetermined threshold. A detailed description of the feature selection process can be found in [31].

We used the Reuters Corpus (RCV1-v2) [29] to generate a set of data streams. RCV1-v2 consists of 804414 news stories, produced by Reuters between August 20, 1996, and August 19, 1997. Each story has been categorized according to its content, and identified by a unique document id.

RCV1-v2 has been processed by Lewis et al. [22]. Features were extracted from the documents, and indexed. A total of 47236 features were extracted. Each document is represented as a vector of the features it contains. We refer to these vectors as feature vectors. We simulate ten streams

by arranging the feature vectors in ascending order (according to their document id), and selecting feature vectors for the streams in a Round Robin fashion.

In the original corpus each document may be labeled as belonging to several categories. The most frequent category documents are labeled with is "CCAT" (the "CORPORATE/INDUSTRIAL" category). In the experiments our goal is to select features that are most relevant to the "CCAT" category.

Each node holds a sliding window containing the last 6000 documents it received (this is roughly the amount of documents received in a month). We chose three features that display different characteristic behavior. The chosen features are "bosnia", "ipo", and "febru". Figure 8 depicts how the chi-square score for each feature evolves over the streams. The chi-square score for the feature "bosnia" displays a declining trend as the stream evolves. The chi-square score for the feature "ipo" remains relatively steady, while the score for "febru" peaks halfway through the stream.

We monitored each feature using threshold values ranging from 30 to 480. We repeated this experiment using the following constraints: (1) The spherical constraints with the estimate vector as the reference vector (the constraints presented in [30]), (2) Ellipsoidal constraint with the estimate vector as the reference vector, (3) Spherical constraint with the internal vector as the reference vector, (4) Ellipsoidal constraint with the internal vector as the reference vector, (5) Theoretic optimal constraint.

The results of these experiments are presented in Figure 9. Since the chi-square scores for the feature "bosnia" and "ipo" remain relatively high (above about 140), all the constraints are more efficient when monitoring these features using low threshold values. The chi-square scores for "ipo" fluctuate around 250, which explains why the communication expenditure is highest for a threshold value of 250. The chi-square score for "bosnia" is more varied, therefore we do not see a distinct decline in communication expenditure. The gap between the performance of the spherical constraints presented in [30] and that of the theoretic optimal constraints is typically two orders of magnitude

With the exception of the local peak about halfway through the stream, the feature "febru" receives a low chi-square score. As a result, the number of times the chi-square score for "febru" crosses a given threshold is low in comparison to the other features, and the gap between the performance of the spherical constraints presented in [30] and the theoretic optimal constraints is typically only one order of magnitude.

We can see that using ellipsoidal bounding regions (with the estimate vector as the reference vector) reduces the communication expenditure by a constant factor in comparison to the spherical constraints. In contrast, using spherical constraints with an internal reference vector is more effective for lower threshold values. This is because the threshold surfaces induced by chi-square and the examined threshold value are such that they allow plenty of space for distancing the reference vector when the chi-square score of the estimate vector is above the threshold value, and very little space when it is below the threshold. When higher threshold values are used, it is more common for the chi-score of the estimate vector to be below the threshold value, thus reducing the effectiveness of the internal reference vector.

The experiments clearly indicate that using constraints that combine ellipsoidal bounds with an internal reference

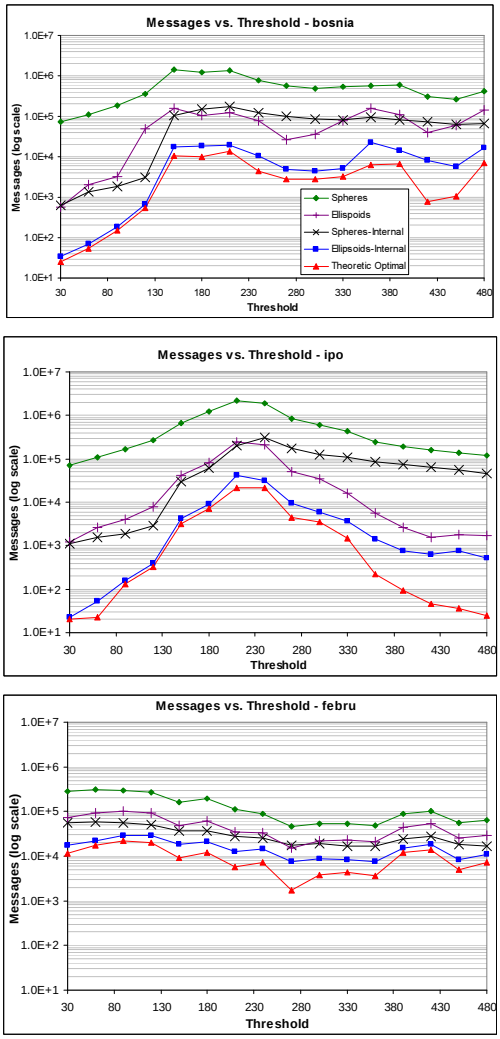


Figure 9: The number of messages generated by monitoring the chi-square score of the features using various constraints.

vector consistently outperform the rest of the constraints. Furthermore, in most cases, the communication cost incurred when using these constraints is close (typically by 50 percent for "bosnia" and "ipo", and by 90 percent for "febru") to the communication cost incurred when using the theoretic optimal constraint.

In all cases, the constraints combining ellipsoidal bounds and an internal reference vector drastically reduced the gap between the communication cost incurred when using the constraints presented in [30] and the cost incurred when using the theoretic optimal constraints.

8. CONCLUSION AND FUTURE WORK

We presented geometric constraints that take advantage of the distribution of the data vectors and the shape of the threshold surface of the monitored function. Using these constraints yields a typical improvement of two orders of magnitude in comparison to the results achieved with previously used constraints. In all cases, the new constraints drastically reduced the gap between previous results, and

those achieved when using theoretic optimal constraints.

Future research will attempt to devise new types of geometric constraints. We also plan to explore alternative methods for resolving constraint violations.

9. REFERENCES

[1] Shipra Agrawal, Supratim Deb, K. V. M. Naidu, and Rajeev Rastogi. Efficient detection of distributed constraint violations. In *ICDE '07*, pages 1320–1324.

[2] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *STOC '96*, pages 20–29.

[3] Arvind Arasu and Gurmeet Singh Manku. Approximate counts and quantiles over sliding windows. In *PODS '04*, pages 286–296.

[4] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *PODS '02*, pages 1–16.

[5] Brian Babcock and Chris Olston. Distributed top-k monitoring. In *SIGMOD '03*, pages 28–39.

[6] Donald Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Greg Seidman, Michael Stonebraker, Nesime Tatbul, and Stanley B. Zdonik. Monitoring streams - a new class of data management applications. In *VLDB '02*, pages 215–226.

[7] Amit Chakrabarti, Graham Cormode, and Andrew McGregor. A near-optimal algorithm for computing the entropy of a stream. In *SODA '07*.

[8] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *ICALP '02*, pages 693–703.

[9] Edith Cohen and Martin J. Strauss. Maintaining time-decaying stream aggregates. *J. Algorithms*, 59(1):19–36.

[10] G. Cormode, R. Keralapura, and J. Ramimirtham. Communication-efficient distributed monitoring of thresholded counts. In *SIGMOD '06*.

[11] Graham Cormode and Minos Garofalakis. Sketching streams through the net: distributed approximate query tracking. In *VLDB '05*, pages 13–24.

[12] Graham Cormode, Minos Garofalakis, S. Muthukrishnan, and Rajeev Rastogi. Holistic aggregates in a networked world: distributed tracking of approximate quantiles. In *SIGMOD '05*, pages 25–36.

[13] Graham Cormode, S. Muthukrishnan, and Wei Zhuang. Conquering the divide: Continuous clustering of distributed data streams. In *ICDE '07*, pages 1036–1045.

[14] Graham Cormode, S. Muthukrishnan, and Wei Zhuang. What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *ICDE '06*, page 57.

[15] Abhinandan Das, Sumit Ganguly, Minos Garofalakis, and Rajeev Rastogi. Distributed set-expression cardinality estimation. In *VLDB '04*, pages 312–323.

[16] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows: (extended abstract). In *SODA '02*, pages 635–644.

[17] Mark Dilman and Danny Raz. Efficient reactive monitoring. In *INFOCOM '01*, pages 1012–1019.

[18] Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. In *SCG '05*, pages 142–149.

[19] Ling Huang, Minos Garofalakis, Joseph Hellerstein, Anthony Joseph, and Nina Taft. Toward sophisticated detection with distributed triggers. In *MineNet '06*, pages 311–316.

[20] Ling Huang, XuanLong Nguyen, Minos N. Garofalakis, Joseph M. Hellerstein, Michael I. Jordan, Anthony D. Joseph, and Nina Taft. Communication-efficient online detection of network-wide anomalies. In *INFOCOM '07*, pages 134–142.

[21] Ankur Jain, Joseph M. Hellerstein, Sylvia Ratnasamy, and David Wetherall. A wakeup call for internet monitoring systems: The case for distributed triggers. In *Proc. 3rd ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, 2004.

[22] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[23] Samuel Madden and Michael J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *ICDE '02*, page 555.

[24] Samuel Madden, Mehul Shah, Joseph M. Hellerstein, and Vijayshankar Raman. Continuously adaptive continuous queries over streams. In *SIGMOD '02*, pages 49–60.

[25] Amit Manjhi, Vladislav Shkapenyuk, Kedar Dhamdhere, and Christopher Olston. Finding (recently) frequent items in distributed data streams. In *ICDE '05*, pages 767–778.

[26] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *VLDB '02*, pages 346–357.

[27] Chris Olston, Jing Jiang, and Jennifer Widom. Adaptive filters for continuous queries over distributed data streams. In *SIGMOD '03*, pages 563–574.

[28] P.A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2):293–320, 2003.

[29] T.G. Rose, M. Stevenson, and M. Whitehead. The Reuters Corpus Volume 1 - from Yesterday's News to Tomorrow's Language Resources. In *LREC '02*, pages 827–832.

[30] Izchak Sharfman, Assaf Schuster, and Daniel Keren. A geometric approach to monitoring threshold functions over distributed data streams. In *SIGMOD '06*, pages 301–312.

[31] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *ICML '97*, pages 412–420.

[32] Byoung-Kee Yi, Nikolaos Sidiropoulos, Theodore Johnson, H. V. Jagadish, Christos Faloutsos, and Alexandros Biliris. Online data mining for co-evolving time sequences. In *ICDE '00*, page 13.

[33] Yonggang Jerry Zhao, Ramesh Govindan, and Deborah Estrin. Computing aggregates for monitoring wireless sensor networks. In *SNPA 03*.

[34] Yunyue Zhu and Dennis Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB '02*, pages 358–369.