Shape Sensitive Geometric Monitoring*

Daniel Keren†
Computer Science Department
Haifa University
Haifa 31905, Israel

Izchak Sharfman
Assaf Schuster
Avishay Livne
Computer Science Faculty
Technion
Haifa 32000, Israel

Index terms: Data Streams, Distributed Systems, Geometric Monitoring, Shape, Data Modeling.

Abstract

An important problem in distributed, dynamic databases is to continuously monitor the value of a function defined on the nodes, and check that it satisfies some threshold constraint. We introduce a monitoring method, based on a geometric interpretation of the problem, which enables to define local constraints at the nodes. It is guaranteed that as long as none of these constraints is violated, the value of the function did 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 data, here we incorporate their temporal behavior. Also, the new constraints are tailored to the geometric properties of the specific monitored function. In addition, we extend the concept of *safe zones* for the monitoring problem, and show that previous work on geometric monitoring is a special case of the proposed extension.

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

1 Introduction

Many emerging applications require processing high-volume streams of data. Examples include network traffic monitoring systems, real-time analysis of financial data [36, 38], distributed intrusion

*A preliminary version of this work appeared in PODS 2008.

†Corresponding author, dkeren@cs.haifa.ac.il.

detection systems, and sensor networks [26]. A key difference between these problems and those handled by traditional DBMS (Database Management System) is that the abovementioned application 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. Here, on the other hand, we are interested in handling 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 [27], various types of aggregation queries [4, 29], and monitoring queries [9]. 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 [26], network monitoring [20], and distributed intrusion detection.

In many cases, the user of a distributed dynamic 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 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 attack, at least one of the agents will detect that the network traffic to that node exceeds 0.1 percent of the incoming traffic. 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 [20].

In more complex monitoring tasks, global "events of interest" may be harder to detect by solely examining local data. For example, 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 terms is a search query. To achieve this, 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 defined by:

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

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

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. A typical requirement in data mining applications is to alert when the correlation coefficient crosses a given positive threshold, since this means that two search terms became correlated. It is easy to see that it's impossible to reach a correct decision, given only the local values of the correlation coefficient. This is a characteristic of non-linear functions; see for example Section 4 ("Running Example") in [33].

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. For the above example, this means that we're not interested in the exact value of the correlation coefficient between the appearances of two tokens A, B at every point in time – the large part of this information being redundant – but only on whether the correlation coefficient had passed a certain threshold. We refer to such tasks as *non-linear monitoring tasks*.

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 vulnerable to a high communication load, since communication is the primary factor affecting the power consumption of the network [37]. In addition to communication load concerns, collecting data to a central location may violate privacy requirements in certain applications.

Previous work [34] 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 then needs to verify 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, it is guaranteed that the function did not cross the threshold. The constraints proposed in [34], 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. Here we

solve this problem by fitting a probabilistic model to the data, and using 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 the 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. To solve this problem, we modify the constraints by tailoring them to the geometric properties of the specific function which is being monitored.

Lastly, we define a general notion of safe zones (SZs). A node's SZ consists of the set of vectors which satisfy the local constraints, and as long as the vectors remain in their SZs, no communication is required. We show that the SZs defined in previous work on geometric monitoring are a special case of a general paradigm for computing optimal SZs, which are defined as certain convex subsets of the function's domain.

2 Related Work

A well studied problem is the monitoring of frequency counts over a single data stream [4, 6, 11, 29], however these works do not address distributed environments. Other important problems over a single data stream were studied in [10, 12, 19, 21].

Distributed function computation has been addressed by the "Distributed Triggers" framework presented in [24]. Later, this framework has been employed to monitor network wide traffic anomalies [22, 23]. 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 [22, 23], which focus on an anomaly detection problem, our work addresses a wide class of non-linear monitoring problems. Recent work addressed thresholding a function defined on distributed nodes in the static case by first solving the problem for a monotonic function and then extending to a general function by expressing it as the difference of two monotonic functions [39].

Another important problem is the computation of the sum or average of a distributed set of variables. Prominent examples include [20], which addresses the problem of detecting when the sum exceeds a given threshold, [13], which proposes observing the distribution of the input data to derive optimal algorithms, and [30] 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 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 bounds. This approach has been employed to detect "heavy hitters" [28], compute quantiles [15], count distinct elements [17], and compute join aggregates [14].

Other distributed computation problems studied in previous work include top-k problems [8], set-expression cardinality estimation [18], clustering [16], and distributed verification of logical expressions [3].

Recent work addresses a theoretical tracking of frequency moments tracking, where the goal is to minimize communication while maintaining an accurate estimate [40], tracking of "heavy hitters" (elements whose frequency is at least a certain portion of a given set) and quantiles [41], and optimal sampling from distributed streams [42]. Recently, the geometric monitoring scheme was proposed in [34]. In contrast to the methods proposed in [34], 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 a very significant reduction in communications.

3 The Monitoring Framework

We now present a general framework for performing non-linear monitoring tasks. 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 these vectors by d (e.g. for the correlation coefficient example, $d = 3$). 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 (below) the threshold are colored white (grey). Given this 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 when 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. it 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. 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.

Fig. 1 (left) 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).

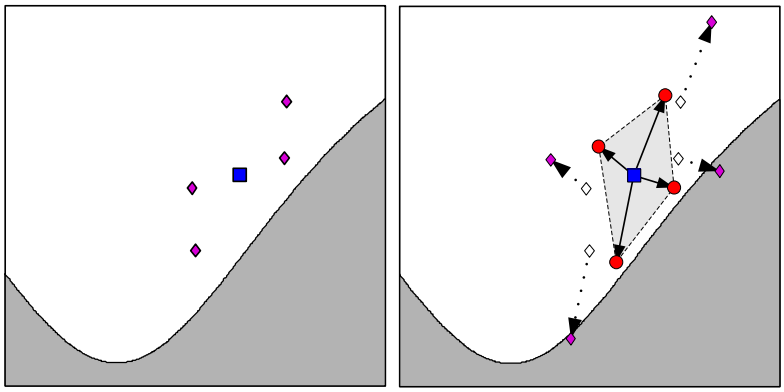


Figure 1: Geometric interpretation of a monitoring problem.

As data arrives at the nodes, the local vectors change. The new location of the data vectors (purple diamonds), as well as their initial location (white diamonds) are depicted in Fig. 1 (right). 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 define local constraints on the values of the drift vectors, such that each node can verify its local constraint independently, such that 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 this convex hull remains monochromatic, the function's value did not cross the threshold and no communication is required.

The local constraints are defined by regions in $\mathcal{R}^d$ 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 in advance on a $d \times d$ symmetric positive definite matrix A, which we name the *shape matrix*. In addition, the nodes agree on a common vector denoted by $\vec{r}$, which is called the reference vector. The shape 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 shape matrix A, the bounding region constructed by the node is the ellipsoid E_A, which is defined as follows, using the Mahalanobis distance [43]:

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

Note that by setting the shape matrix to the unit matrix, the bounding region maintained by each

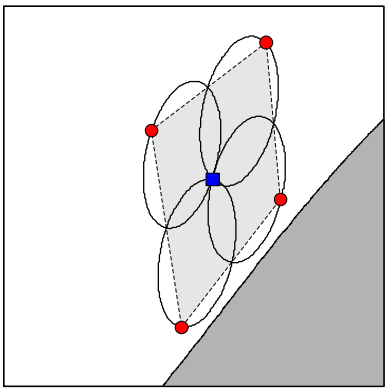


Figure 2: The use of ellipsoidal constraints.

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 the identity matrix as the shape matrix by $B(\vec{r}, \vec{v}_i)$:

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

Using the identity matrix as the shape matrix and the estimate vector as the reference vector yields the spherical bounds defined in [34].

When the shape matrix is not the identity, the bounding region held by a node is an ellipsoid centered at the midpoint between the reference vector and the node's drift vector. Fig. 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 shown. As illustrated in the figure, all the ellipsoids are monochromatic, guaranteeing that the convex hull is monochromatic as well (note that the covering theorem is correct for every choice of A). For a proof of the general case, see Section 4.2.

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 shape 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 shape 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 adjust 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 shape matrix.

4 Constructing Data Sensitive Constraints

The goal in this section is to construct bounding regions that cover the convex hull of the drift vectors as tightly as possible. We thus decrease the number of "false positives" generated by the constraints. A constraint causes a "false positive" if the bounding region it defines is not monochromatic, while the convex hull of the drift vectors is monochromatic; that means that the alert sent by the node was spurious. Intuitively, we want the bounding regions to be "large" – not necessarily in terms of volume, but in terms of the probability of the dynamic local vectors to remain in their bounding volumes for as long as possible. While these optimal bounding volumes are not required for *correctness*, they substantially improve the monitoring algorithm by minimizing communication.

We use previous data received on the streams to construct a probabilistic model of future values, and then use this model to construct optimal 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 (normal) distribution, as the number of search queries held in the sliding windows increases. Recall the multi-variate Gaussian distribution:

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

Where $\vec{\mu}$ 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 construct an empirical probability distribution of

these values by calculating their mean and covariance matrix, and use them as the parameters of a multi-variate Gaussian distribution. Then, 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. We show that by first applying this transformation on the drift vectors, and then using spherical bounding regions as described in Section 3, we obtain 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 obtain the tightest possible ellipsoidal bounding regions.

We start by describing the normalization process. Given $G_{\vec{\mu},\mathbf{\Sigma}}(\vec{v})$, 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 $T_\Sigma = D'^{-1}_\Sigma P_\Sigma^{-1}$. 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 (in the pattern recognition community such an operation is often referred to as "whitening", [43]).

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 Eq. 1). According to a theorem from [34] (which is quoted below), it 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* $\vec{x}, \vec{y}_1, \vec{y}_2, ..., \vec{y}_n \in \mathcal{R}^d$. *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 $E_{\Sigma^{-1}}(\vec{e}, \vec{v}_1)$,...,$E_{\Sigma^{-1}}(\vec{e}, \vec{v}_n)$. Fig. 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).

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 shape 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 (see Section 8).

Next, we formally show that using the inverse of the covariance matrix produces optimal ellipsoidal bounding regions.

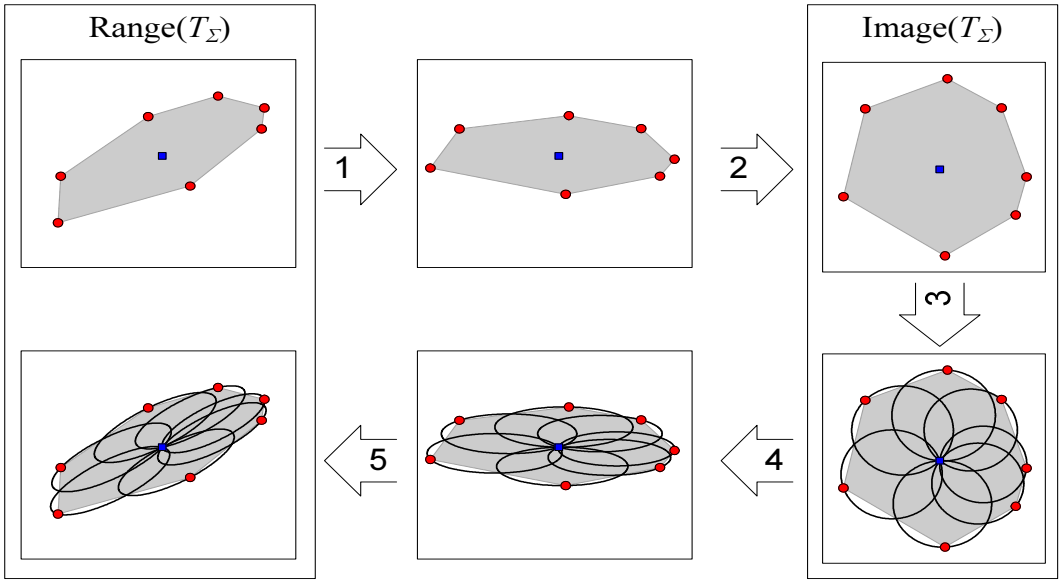


Figure 3: Bounding a convex hull with ellipsoids.

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 shape 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 shape matrix, we should receive ellipsoids whose expected volume is greater. Theorem 2 formally confirms this intuition; due to space limitations, its proof is not included.

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 $A = \frac{\mathbf{\Sigma}^{-1}}{\det(\mathbf{\Sigma}^{-1})^{1/d}}$.*

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 needs to be bounded, but 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.

Fig. 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 (red circles) ((0.45,0.39) and (0.105,0.055)) and an estimate vector (blue square) (0.26,0.268). The figure depicts two choices of reference vectors: in Fig. 4 (left) the estimate vector is used as the reference vector, while in Fig. 4 (right), 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.

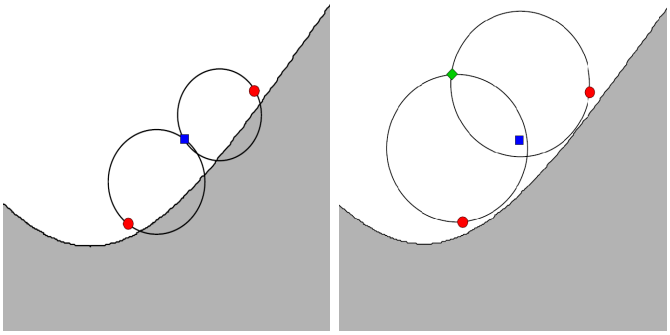


Figure 4: The effect of the reference vector on the bounding regions. The green reference point, obtained by translating the blue reference point away from the white region's boundary, yields larger bounding volumes for the two red dots representing data points; however, these volumes are contained in the white region, while the smaller ones, dictated by the blue reference point, are not.

Next we will expand upon the intuitive concepts described above. We begin by describing some concepts, then define a formal construct called a safe zone. Then, 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 $\mathcal{R}^d$.

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 $\text{dist}(T(g,t),\vec{x})$:

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

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 Fig. 5.

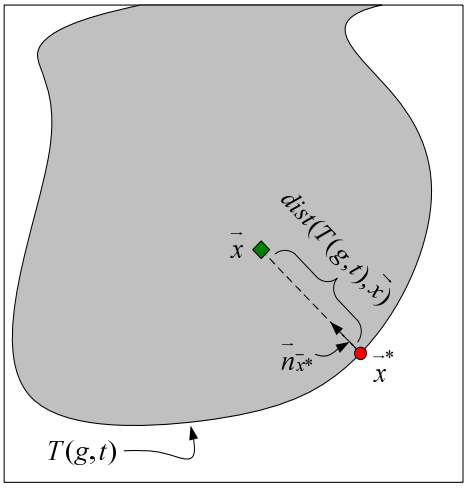


Figure 5: Illustration of a threshold surface, a reference point, and the normal.

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 $\vec{r}$:

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

We call this set the *safe zone* induced by g, t, and $\vec{r}$, indicating that as long as the drift vectors lie inside their safe zones, it is guaranteed that the function did not cross the threshold, and no communication is required.

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. Fig. 6 illustrates the safe zone defined by the estimate vector and threshold surface that were depicted in Fig. 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, 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

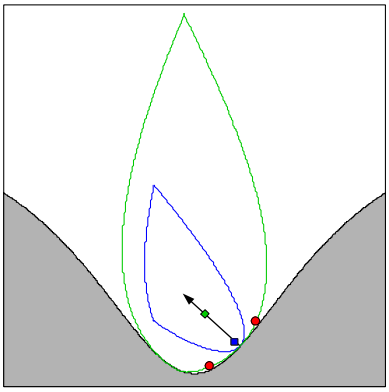


Figure 6: The estimate vector (blue) and the distant vector (green) that were depicted in Fig. 4, and their respective safe zones.

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 the safe zone 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. 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 $[S_{g,t}(\vec{r}_1) \cap L] \subset [S_{g,t}(\vec{r}_2) \cap L]$. Note that this definition is with respect to a specific 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:

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 $\vec{r} \in R$, $\mathrm{col}_{g,t}(\vec{r}) = \mathrm{col}_{g,t}(\vec{e})$.

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 $\vec{r}_{max}$, whose radius is the distance between $\vec{r}_{max}$ and $\vec{e^*}$ (see Fig. 7). Since $\vec{e}$ is in the line segment connecting $\vec{r}_{max}$ and $\vec{e^*}$, 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: We need to show that for every vector $\vec{z} \in L_{max}$, the sphere $B(\vec{r}_{max}, \vec{z})$ is monochromatic.

[2]The term "connected set" refers to a pathwise 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 interval [0,1] into the set such that $f(0) = \vec{x}$, $f(1) = \vec{y}$.

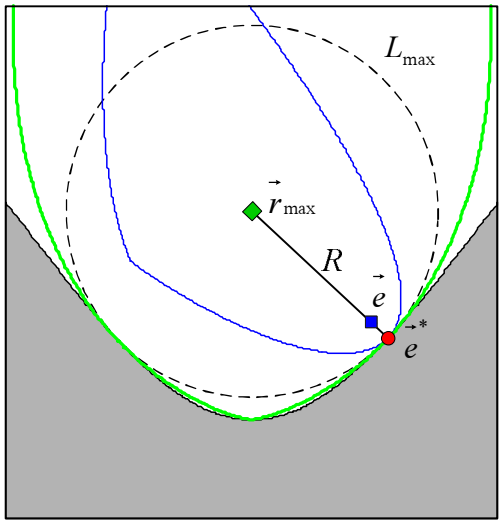


Figure 7: A threshold surface, an estimate vector, the set R and the vicinity L_{max}.

It is easy to see that this sphere is contained in L_{max}, therefore it is sufficient to show that L_{max} is monochromatic. Let us assume, by way of contradiction, that L_{max} is not monochromatic, i.e. there is a vector $\vec{z} \in L_{max}$ such that $\mathrm{col}_{g,t}(\vec{z}) \cdot \mathrm{col}_{g,t}(\vec{r}_{max}) = -1$. Since g is continuous, there exists a vector $\vec{z'}$ on the line segment between $\vec{r}_{max}$ and $\vec{z}$ such that $\mathrm{col}_{g,t}(\vec{z'}) = 0$, i.e. $\vec{z'}$ is on the threshold surface. Note that $\left\|\vec{z'} - \vec{r}_{max}\right\|_2 < \|\vec{z} - \vec{r}_{max}\|_2 \leq \left\|\vec{e^*} - \vec{r}_{max}\right\|_2$. In summary, $\vec{z'}$ is a vector on the threshold surface that is closer to $\vec{r}_{max}$ than $\vec{e^*}$. This stands in contradiction to $\vec{e^*}$ being the vector on the threshold surface that is the closest to $\vec{r}_{max}$, thus concluding the proof.

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: Given $\vec{x}$, let us search for the vector $\vec{y}$ on the threshold surface that is closest to $\vec{x}$. Using the terminology and results of Lagrange multiplier theory, denote $F(\vec{y}) = \|\vec{y} - \vec{x}\|_2^2 + \lambda(g(\vec{y}) - t)$. The sought $\vec{y}$ satisfies $\frac{\partial F}{\partial \vec{y}} = 2(\vec{y} - \vec{x}) + \lambda(\nabla g)(\vec{y}) = 0$, hence $(\nabla g)(\vec{y}) = -\frac{2}{\lambda}(\vec{y} - \vec{x})$. However it is well known from calculus that $(\nabla g)(\vec{y})$ is parallel to the normal to the threshold surface at $\vec{y}$, thus concluding the proof.

The fact that the set R is one-dimensional 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 [31]). 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 (Section 8) show that applying 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 [34].

5.3 Selecting Reference Vectors while Employing Ellipsoidal Bounding Regions

In Section 4 we leveraged a probabilistic model of the data to reduce communication load by employing ellipsoidal constraints. In Section 5.2 we reduced communication by selecting a better reference vector, but we used spherical constraints. 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 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 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 [34], and in certain cases exceeds three orders of magnitude.

5.4 An Example

In some cases, it is possible to provide an accurate description of the safe zone, and the optimal choice of the reference vector. We now present an example of a very simple but illustrative threshold surface. Let the data be two-dimensional, (x, y), the monitored function equal to y, and the threshold equal to 0; so, the allowable region G is simply the upper half-plane. It's relatively simple to compute the safe zone associated with a point.

Lemma 3 *The safe zone (using spherical constraints) corresponding to a point* $r_0 = (x_0, y_0)$ *is the parabola* $y = \frac{(x-x_0)^2}{4y_0}$.

proof: Let D be the disk whose diameter is the line segment connecting r_0 and a point $p = (x, y)$. For the disk to be contained in the upper half-plane, it is enough to check whether its lowest point is in the upper half-plane; but the y coordinate of the lowest point is $\frac{y+y_0}{2} - \frac{||p-r_0||}{2}$. The result follows immediately.

Assume that the data distribution is a Gaussian centered at $(0, 1)$, with an eccentricity of 4, and at an angle of $\frac{\pi}{4}$ (see Fig. 8). The equation of this Gaussian is $e^{-\frac{17}{4}x^2+\frac{15}{2}xy-\frac{15}{2}x-\frac{17}{4}y^2+\frac{17}{2}y-\frac{17}{4}}$. In order to illustrate the considerations affecting the choice of a better reference point, we now show an incremental version (as opposed to the global choice of a single best reference point, treated in Section 5.3). The question posed is: given a current reference point, how should it be perturbed in order to obtain an optimal reference point? If we fix the length of the perturbation to some small positive constant (taken to be 0.1 in the following example), the question is in what direction to move in order to maximize the total probability in the safe zone. To determine this, the following iterative computation was performed: at each stage, the integral of the Gaussian was computed for the safe zones of all points at a distance of 0.1 from the current reference point, and the one with the highest integral was chosen as the new reference point. The path that results from concatenating these points (Fig. 8) represents the optimal path by which to move away from G's boundary. Note that for an axes-aligned Gaussian, it will be simply a straight vertical line, resulting in paraboloid regions with increasing areas; here however, the safe zones are chosen so as to also contain as much "probability volume" of the Gaussian as possible.

6 Theoretic Optimal Geometric Constraint

Experimental results (Section 8) show that employing the new geometric constraints presented above very substantially reduces communication. 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

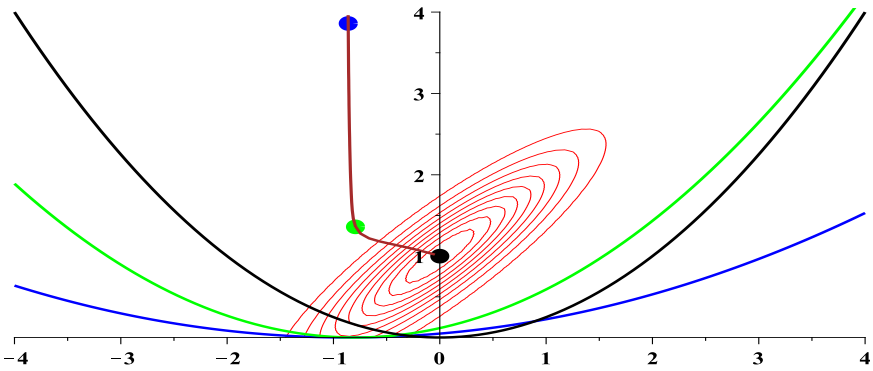


Figure 8: The optimal path (brown curve) by which to move the original reference point (dark circle) away from G's boundary (the x-axis). The red ellipses depict the level sets of the Gaussian distribution representing the data, and the three parabolic curves correspond to the safe zones of the reference points with the respective colors. Note that initially the curve does not move vertically away from the boundary (as it would for an isotropic Gaussian), but also curves to the left, in order to "capture more probability" of the Gaussian inside the safe zone (since this results in the apex of the parabola moving to the left as well).

the only thing 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 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 is at all possible. In Section 8 we compare results not only to previous work but to the theoretic optimal constraints as well.

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 [35].

We used the Reuters Corpus (RCV1-v2) [32] 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. [25]. 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 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 this 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". Fig. 9 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 score for "ipo" remains relatively steady, while the score for "febru" peaks halfway through the stream.

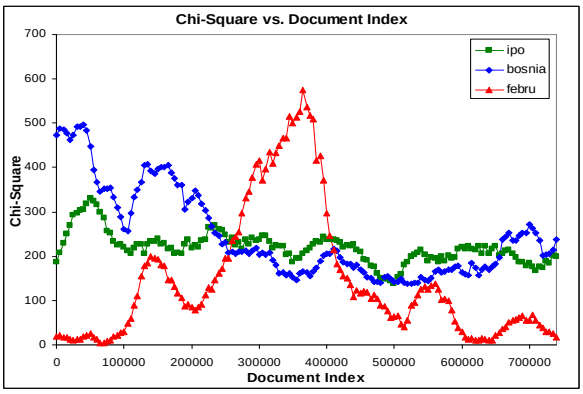
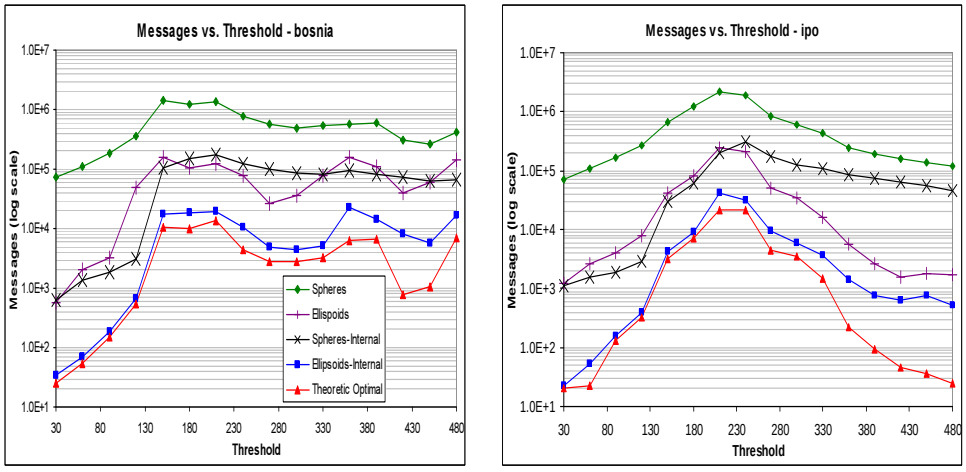


Figure 9: Chi-square score for the features "Bosnia", "ipo", and "febru" as it evolves over the streams.

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 [34]), (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 Fig. 11. Since the chi-square scores for "Bosnia" and "ipo" remain relatively high (above 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 [34] 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, "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 [34] and the theoretic optimal constraints is typically only one order of magnitude.



It can be observed 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

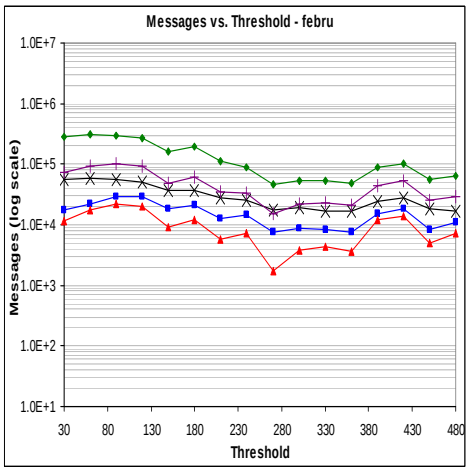


Figure 10: The number of messages generated by monitoring the chi-square score of the features using various constraints.

chi-square 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 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.

8 Generalization: Safe Zones as Convex Subsets of G

We now present a more general view of the monitoring problem, which includes the safe zones defined by spherical/ellipsoidal bounding volumes as a special case. The idea behind this generalization commences with the observation that the safe zones defined by the spherical bounding volumes are always convex, and vice-versa (every convex subset of the allowable region can serve as a safe zone). Therefore, one may as well search for the *optimal* convex subset. We now proceed to formalize and prove the relevant concepts and theorems, and to compare the generalized approach with the one presented in the previous sections.

As before, denote the allowable region, defined by the monitored function $g(x)$ and the threshold t, by $G = \{x|g(x) < t\}$, and let the reference point be denoted by r. Recall that the safe zone S consists of all the points v such that $B(v, r) \subset G$. The following lemma concerns a general property of S, which holds for every function, threshold, and reference point:

Lemma 4 *S is convex.*

Proof: first, note that $B(v,r) = \{x|(x-v, x-r) \leq 0\}$ (this is straightforward; in two dimensions, by the way, it follows immediately from *Thales' theorem*, which states that an inscribed angle in a semicircle is a right angle). Now, assume that $\{v_i\}_{i=1}^n$ are in S – that is, for $1 \leq i \leq n$, $B(v_i, r) \subset G$. Now let v be a convex combination of v_i, so there are scalars λ_i such that $\lambda_i \geq 0$, $\sum_i \lambda_i = 1$, $\sum_i \lambda_i v_i = v$. To prove that $B(v,r) \subset G$ (hence $v \in S$), we will prove that $B(v,r) \subset \cup_i B(v_i, r)$. Assume $x \in B(v,r)$; so $(x-v, x-r) \leq 0$. Since $\sum \lambda_i = 1$, $(x-v, x-r) = (\sum \lambda_i x - \sum \lambda_i v_i, x-r) = \sum \lambda_i (x-v_i, x-r)$. If for all i, $x \notin S_i$, then for all i $(x-v_i, x-r) > 0$, and since λ_i are positive, this would imply that $\sum \lambda_i (x-v_i, x-r)$ is also positive; but this last expression equals $(x-v, x-r)$ which we know to be negative. So, there must be an i such that $x \in S_i$, which concludes the proof. Note that this will also hold for ellipsoidal constraints. As the following lemma demonstrates, the fact that the safe zones defined by the spherical constraints is a convex subset of G can be viewed as a special case of a sufficient condition for safe zones.

Lemma 5 *Assume WLG that the reference vector is the origin O, and denote the initial data vector at node i by v_i^0 (note that $\sum_i v_i^0 = 0$, since the reference vector is the average of the local vectors). Let C be any convex subset of G which contains O. Then the sets defined by $S_i(C) \triangleq \{v_i^0 + c | c \in C\}$ are legal safe zones at the nodes. Note that the safe zone at the i-th node is the translation of C by the node's initial data vector.*

Proof: we need to prove that as long as the local vectors are in their respective safe zones, the threshold of the monitored function was not crossed – that is, that the global vector did not wander out of G. Let the local vector at node i equal $v_i^0 + c_i$, for some $c_i \in C$. Then the global vector is the average of these local vectors which equals $g \triangleq \frac{\sum_i c_i}{n}$ (since $\sum_i v_i^0 = 0$). But since C is convex, $g \in C$, hence $g \in G$ (recall that C is a subset of G).

8.1 Safe Zones Defined by Convexity

As Lemma 5 suggests, we can define safe zones by the following construct. Assume that the probability distribution of the monitored data (which in the previous sections we modeled as a Gaussian distribution) is $p(x)$. As before, assume WLG that the reference vector is the origin O. Then, solve the following optimization problem:

$$\text{maximize} \int_C p(x)dx \quad \text{such that } C \text{ is a convex subset of } G \text{ and } O \in C. \tag{4}$$

Denoting the optimal C by C_0, assign the i-th node the safe zone defined by C_0 translated by v_i^0. These safe zones are legal, since as proved in Lemma 5 the average of vectors in the safe zones is inside G. Maximizing the overall probability for the data to be in C_0 guarantees that the local vectors will remain in their safe zones for the longest duration possible. Note that if one does not impose convexity, the resulting safe zones will not be valid, since the average of vectors which belong to a non-convex set need not be in the set.

8.2 Computing Convex Safe Zones

In order to find the optimal convex safe zone, it is required to optimize over all convex subsets of G. Generally, this is impossible, since the class of all such subsets is a non-linear, infinite dimensional space. To define an optimization problem which can be practically solved in the general case, one must restrict attention to a finitely-parameterizable family of subsets. In two-dimensional Euclidean space $\mathcal{R}^2$, a natural choice are convex polygons, which can be defined as the convex hull of n points. Thus, there will be $2n$ parameters to the optimization problem (the coordinates of these n points). The first question which arises is: how much of the safe zone's quality are we losing by restricting it to a convex polygon with n vertices? The following theorem [2] provides an answer:

Theorem 3 *let C be a convex subset of the plane, and let k be a fixed integer. Denote by C_k the maximal value of the ratio $\frac{A(P_k)}{A(C)}$, where A stands for area and P_k is any convex polygon with k vertices inscribed in C . Then the minimal value of C_k is obtained when C is a disk. It is straightforward to see that as k increases, C_k behaves as $1 - \frac{\alpha}{k^2}$ for some small constant α.*

This theorem quantifies the rate of approximation of an inscribed convex polygon to an *arbitrary* convex set in the plane. In d dimensions, the bound $\frac{\alpha}{k^2}$ is replaced by $\frac{cd}{k^{\frac{2}{d-1}}} < \frac{cd}{k}$ for a dimension-independent constant c. This demonstrates that with a reasonable number of vertices, a very good approximation can be achieved. Thus, it is sufficient to restrict attention to safe zones defined as the convex hull of an appropriately chosen number of points.

8.3 Details of the Optimization: Target Function and Constraints

The optimal convex subset is solved for by a constrained optimization paradigm. The constraints are 1) that it is convex, and 2) that it is contained in G. The target function to maximize under these constraints is $\int_C p(x)dx$. As noted in Section 8.2, convexity is guaranteed by defining the candidate safe zone C as the convex hull of the points which are the input to the optimization routine (we have used Matlab's function *convhull* to construct the convex hull). There are a few options to test whether C is contained in G; typically, G is described by a certain inequality, and standard methods, such as

Lagrange multipliers, can be used to determine the point in C farthest from G. The distance between this point and G is then used in the optimization routine as the measure of constraint violation (we have used Matlab's *fmincon* routine). It remains to calculate the target function, $\int_C p(x)dx$. We tested two methods: the most direct one is integrating the Gaussian p.d.f over C. In the two-dimensional case, we can use Green's theorem to reduce the integral to a one-dimensional integral over C's boundary ∂C, using the identity

$$\iint_C \exp\left(-Ax^2 - 2Bxy - Cy^2 - Dx - Ey - F\right) dxdy =$$
$$\frac{\sqrt{\pi}}{2\sqrt{A}} \oint_{\partial C} \exp\left(-Ey - F - Cy^2 + \frac{(D+2By)^2}{4A}\right) \mathrm{erf}\left(\sqrt{A}x + \frac{D+2By}{2\sqrt{A}}\right)$$

In higher dimensions, one can use Stokes' theorem, or apply numerical integration methods. Another option is to estimate the quality of C directly from the discrete data available at the nodes, before it is approximated by a Gaussian distribution. To implement this method, we defined the target function as $\sum_i \exp\left(-\lambda d(p_i, C)\right)$, where the data points are denoted p_i and $d(p_i, C)$ is the distance from p_i to C (which equals zero if $p_i \in C$), and λ is a positive constant. The motivation for not taking the target function as the number of points inside C is that this defines a discontinuous target function, with jump discontinuities whenever a point enters or exits C. In our experience, optimization routines find it difficult to handle such discontinuities.

Some examples are now provided, in which the two types of safe zones are compared. The data consists of wind measurements of the El-Nino system in two directions (x is for east-west and y for south-north), as taken from [5]. The first function to monitor (Fig. 11) was $xy - 196$, hence G is the set bounded from above by the graph of $y = \frac{196}{x}$, and from below(left) by the positive half of the $x(y)$ axes. Fig. 11 depicts the safe zone defined by spherical bounding volumes and the one found by optimizing over all hexagons contained in G. Here we used a discrete probability model. Next (Fig. 12) we show results for a more complicated G, whose boundary is made both of concave and convex parts. The monitored function is a fourth-degree polynomial, $x^4 + y^4 - 5(x^2 + y^2) + 10x - 3$.

8.4 Another Look at the Safe-Zone Defined by Bounding Volumes

Here we take a closer look at the structure of the safe zones defined by spherical bounding volumes. In addition to providing another proof why the safe zone is convex, it will allow to elucidate on its structure, and compare it to the optimal convex subset defined in Sections 8.2-8.3.

Recall (we'll follow here the notations used in the beginning of Section 7) that for a given G and reference vector r, the safe zone (denote it $S(G, r)$) equals all the points v such that $B(r, v) \subset G$.

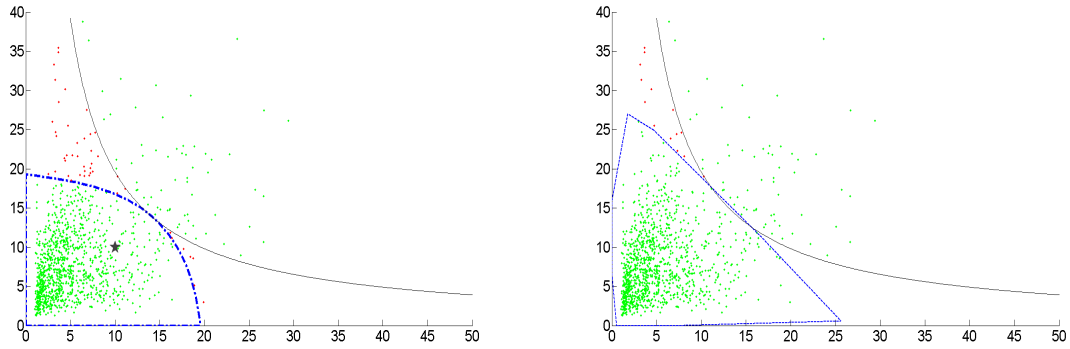


Figure 11: Example of a convex subset of G as safe zone. The monitored function is $xy - 196$, so G is bounded by the hyperbolic curve and the positive axes; data is depicted as green dots, and the reference point by an $\times$. Depicted are the safe-zone defined by spherical bounding regions (top), outlined in blue. Data points correctly classified are marked in green, misclassified in red (note that points outside G and also outside the safe-zone are correctly classified). Bottom: a polygonal safe-zone (also outlined in blue) which was derived by solving the optimization problem that seeks the optimal convex hexagon contained in G. For the decagon, 94.6% of the points were correctly classified, and for the safe zone defined by the bounding volumes, 92.5% were correctly classified.

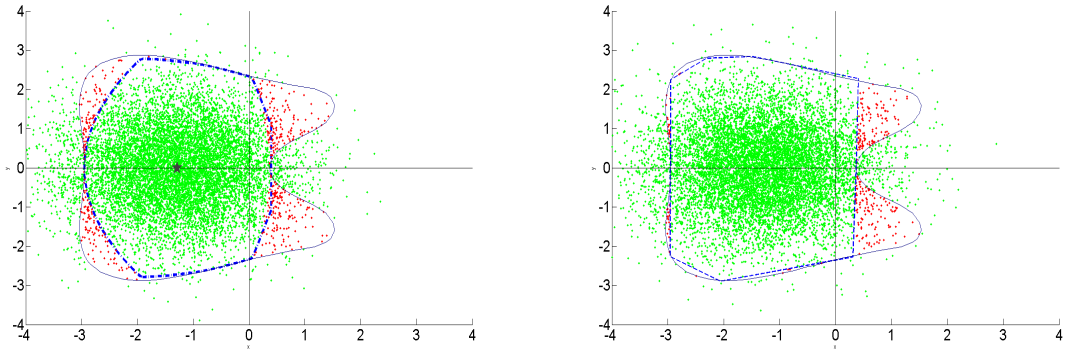


Figure 12: Same as Fig. 11, for a more complicated G. For the optimal convex decagon (bottom), 97.4% of the points were correctly classified, and for the safe zone defined by the bounding volumes (top), 94.9% were correctly classified.

Clearly, $S(G,r) = \{v|B(r,v) \cap G' = \phi\}$, where G' denotes the complement of G. Hence, $S(G,r) = \cap_{y \in G'} A_y$, where $A_y \triangleq \{v|y \notin B(r,v)\}$. It is straightforward to see that A_y is the half-plane passing through y and perpendicular to the segment connecting r and y, and which contains r (Fig. 13). It is also trivial that if $y \notin G$, then a vector $z \notin G$ which satisfies $A_z \subset A_y$ can be found by infinitesimally

translating y towards G. To conclude:

Theorem 4 *: $S(G, r)$ equals the intersection of half-planes supported by G's boundary points, where for each boundary point u the boundary of the respective half-plane passes through u and is perpendicular to the segment connecting r and u (note that the half-plane boundary is generally* **not** *tangent to G's boundary).*

An example is presented in Fig. 13. It can be seen that, while the intersection of the half-planes

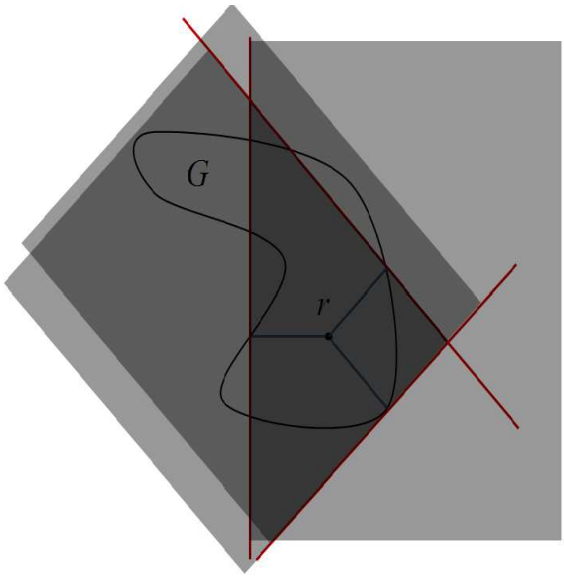


Figure 13: A sketch demonstrating Theorem 4. Depicted are G and a reference point r. The three red lines represent the boundaries of three half-planes, and their intersection is the dark triangle. The safe zone equals the intersection of *all* half-planes supported by G's boundary, with their boundary perpendicular to the line segment between r and the boundary point.

defined a convex set (since a half-plane is convex and any intersection of convex sets is convex), the convex subset of G thus defined is not optimal, since the half-planes often cut "too much" from G.

8.5 Computation and Evaluation of Safe-Zones

There are a few factors which influence the decision whether to use a safe-zone defined by bounding volumes or by optimizing over convex subsets. The obvious advantage of using the bounding volume approach is that it requires no optimization. The convex subset approach, however, requires to solve what may be a non-trivial optimization problem, especially if the data resides in a high-dimensional space. Another factor is the complexity of testing whether the local data vectors are inside their safe-zones or not; for the bounding volume approach, this requires testing whether a sphere is inside

G, and for the convex subset approach, whether a point is inside it. To summarize, the choice of which approach to use depends on the safe-zones' quality (which is measured by the value of target function), difficulty of the optimization problem, and complexity of testing whether the local vectors are in their safe zones.

9 Running Times

We now discuss running times and performance for both algorithms. For the bounding volumes approach, checking whether a local condition has been violated at a certain node requires determining whether the sphere subtended by the local data vector and the reference point is contained in the set $G \triangleq \{x|f(x) \leq T\}$, for the monitored function f and threshold T (in case ellipsoidal bounding volumes are used, G is transformed by the same transformation that renders the ellipsoidal bounding regions spherical). Therefore, the question is how distant a point (the sphere's center) is from $G's$ boundary. The running time for solving this problem was addressed in previous work (see Section 7 in [33]). Here we ran tests on additional data and compared them to the approach applying an optimal convex subset. The data used was air pollution measurements [1]. We used data of up to four dimensions, consisting of measurements of SO2, PM10, O3, and NO, all in micrograms per cubic meter. Typical data for two of the components is depicted in Fig. 14. The global function monitored was a weighted sum of the pollutant concentration, which is used in estimating the AQI (air quality index). The average running time for testing the local conditions was 0.032 seconds for four-dimensional data, 0.021 seconds for three-dimensional data, and 0.0013 seconds for two-dimensional data. These averages were computed for 1,000 runs.

9.1 Comparison Between the Two Algorithms

In terms of running time, there is a difference between the two approaches presented here: the bounding volumes approach requires no pre-processing, as its safe zone is guaranteed to be correct, while the optimal convex subset approach requires to first determine the safe zone. However, typically the running time for checking the local conditions in the optimal convex subset approach is shorter, as the definition of the subset is relatively simple (there is no need to compute the distance of a point from $G's$ boundary, as for the bounding volume method). A typical result is now presented, for two-dimensional data consisting of the SO2 and NO pollutants from the database introduced in the beginning of this section. G is an ellipse with a horizontal axis of length 30 and a vertical axis of length 120, centered at the data's center of gravity. There were a total of 1,500 observations across the 235 nodes used, 178 of which yielded a global violation. The bounding volume approach yielded a total of 223 alerts (meaning that 45 of them were false alarms). The optimal convex subset method was applied to this

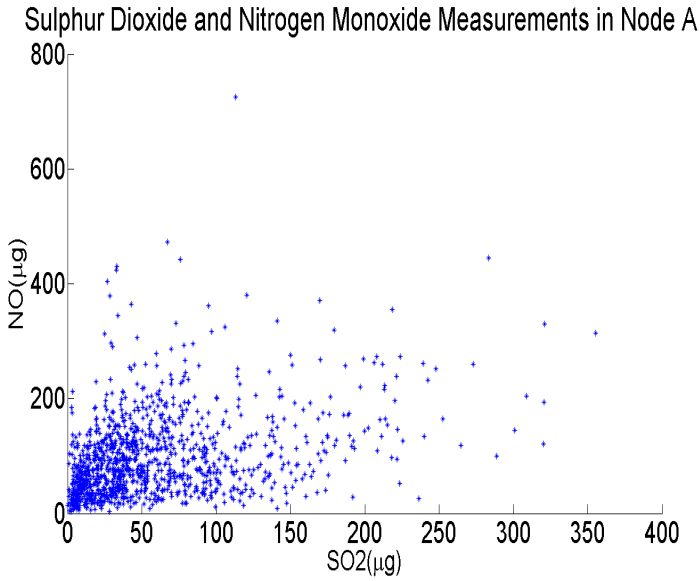


Figure 14: Data for two of the air pollutants used in the experiments.

data, with the safe zones respectively defined as convex polygons with number of vertices ranging from four to ten. Results are presented in the following table:

Number of vertices	Number of alerts	Optimization time (seconds)
4	365	19.22
5	284	53.49
6	270	66.63
7	257	78.79
8	238	87.42
9	199	91.97
10	180	95.88

while the pre-processing time increases with the safe zone's complexity, the richer safe zones yield lower communication overhead. The time for checking the local conditions in this case is negligible (consisting of the time to test whether a point is inside a convex polygon, which is logarithmic in the number of vertices). Note that the pre-processing stage (whose running time is provided in the table above) has to be performed only once.

10 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. We also defined a more general concept of safe zones, based on convex subsets of the function's allowable region. While this approach requires pre-processing in order to determine the safe zone, it can yield lower communication overhead.

Future research will attempt to devise new types of geometric constraints. We also plan to explore methods for resolving constraint violations, and study cases in which the data distributions greatly vary between nodes.

11 Acknowledgment

This submission greatly benefited from the remarks of three anonymous reviewers.

12 Appendix: Common Notations

The following table summarizes the most common notations used throughout this paper.

$f()$	monitored function
T	threshold
G	the set $\{x \vert f(x) \leq T$ ("allowable region")
$E_A(\vec{r}, \cdot)$	bounding volume defined by a reference vector $\vec{r}$ and shape matrix A (Eq. 1)
$G_{\vec{\mu},\mathbf{\Sigma}}$	multi-variate Gaussian distribution with average $\vec{\mu}$ and covariance matrix $\mathbf{\Sigma}$
$p(x)$	data distribution at the nodes

References

[1] The European air quality database http://dataservice.eea.europa.eu/dataservice/metadetails.asp?id=1079

[2] Y. Gordon, M. Meyer, and S. Reisner. Constructing a polytope to approximate a convex body. In *Geometriae Dedicata*, 57:217–222, 1995.

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

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

[5] http://archive.ics.uci.edu/ml/datasets/El+Nino

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

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

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

[9] 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.

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

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

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

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

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

[15] 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.

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

[17] 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.

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

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

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

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

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

[23] 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.

[24] 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.

[25] 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.

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

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

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

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

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

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

[32] 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.

[33] Izchak Sharfman, Assaf Schuster, and Daniel Keren. A geometric approach to monitoring threshold functions over distributed data streams. In *ACM Transactions on Database Systems*, 32.4, 2007.

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

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

[36] 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.

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

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

[39] Guy Sagy, Daniel Keren, Izchak Sharfman and Assaf Schuster. Distributed Threshold Querying of General Functions by a Difference of Monotonic Representation. In *PVLDB 2010*, 4(2), pages 46-57.

[40] Graham Cormode, S. Muthukrishnan and Ke Yi. Algorithms for distributed functional monitoring. In *SODA 2008*, 1076-1085.

[41] Ke Yi and Qin Zhang. Optimal tracking of distributed heavy hitters and quantiles. In *PODS 2009*, 167-174.

[42] Graham Cormode, S. Muthukrishnan, Ke Yi and Qin Zhang. Optimal sampling from distributed streams. In *PODS 2010*, 77-86.

[43] B.V.K Vijaya Kumar, Abhijit Mahalanobis, and Richard D. Juday. Correlation Pattern Recognition. Cambridge University Press, New York, NY, 2010.

13 Author biographies

Daniel Keren (http://cs.haifa.ac.il/~dkeren/, (Ph.D. 1991, Hebrew University in Jerusalem) is with the computer science department in Haifa University, Haifa, Israel. Prof. Keren's main fields of research are geometry and probability. He published mostly in computer vision journals and conferences. Since 2003, he has been working closely with Prof. Assaf Schuster's group in the Technion, in the area of distributed monitoring. His main contribution is in the mathematical aspects of the research such as object modelling, learning, optimization, and probability. A main novelty of the joint research is the incorporation of such mathematical tools into the research paradigm; this allowed to develop new methodologies, based on geometry, to monitor general functions.

Izchak Sharfman recently completed his Ph.D. in the Computer Science Faculty, the Technion. His main area of research is distributed algorithms and geometric methods for stream processing.

Assaf Schuster (http://www.cs.technion.ac.il/~assaf, Ph.D. 1991, Hebrew University in Jerusalem) has established and is managing DSL, the Distributed Systems Laboratory (http://dsl.cs.technion.ac.il). Several CS faculty members see DSL as the main scope hosting their applied and systems research, with about 35 graduate and hundreds of undergraduate students working in the lab during the academic year. DSL is supported by Intel, Microsoft, Sun, IBM, and other interested partners. Prof. Schuster is well known in the area of parallel, distributed, high performance, and grid computing. He published over 160 papers in those areas in high-quality conferences and journals. He regularly participates in program committees for conferences on parallel and distributed computing. He consults the hi-tech industry on related issues and holds seven patents. He serves as an associate editor of the Journal of Parallel and Distributed Computing, and IEEE Transactions on Computers. He supervises seven Ph.D. students and ten M.Sc. students, and takes part in large national and EU projects as an expert on grid and distributed computing.

Avishay Livne obtained his M.Sc degree from the Computer Science Faculty, the Technion. He is currently a graduate student in the University of Michigan, Ann Arbor.