Lightweight Monitoring of Distributed Streams

Arnon Lazerson
Technion – Israel Institute of Technology
Haifa 32000 Israel
lazerson@cs.technion.ac.il

Daniel Keren
Haifa University
Haifa 31905 Israel
dkeren@cs.haifa.ac.il

Assaf Schuster
Technion – Israel Institute of Technology
Haifa 32000 Israel
assaf@cs.technion.ac.il

ABSTRACT

As data becomes dynamic, large, and distributed, there is increasing demand for what have become known as *distributed stream algorithms*. Since continuously collecting the data to a central server and processing it there incurs very high communication and computation complexities, it is advantageous to define *local* conditions at the nodes, such that – as long as they are maintained – some desirable *global* condition holds.

A generic algorithm which proved very useful for reducing communication in distributed streaming environments is *geometric monitoring* (GM). Alas, applying GM to many important tasks is computationally very demanding, as it requires solving a notoriously difficult problem – computing the distance between a point and a surface, which is often very time-consuming even in low dimensions. Thus, while useful for reducing communication, GM often suffers from exceedingly heavy computational burden at the nodes, which renders it very problematic to apply, especially for "thin", battery-operated sensors, which are prevalent in numerous applications, including the "Internet of Things" paradigm.

Here we propose a very different approach, designated CB (for Convex/Concave Bounds). CB is based on directly bounding the monitored function by suitably chosen convex and concave functions, that naturally enable monitoring distributed streams. These functions can be checked on the fly, yielding far simpler local conditions than those applied by GM. CB's superiority over GM is demonstrated in reducing computational complexity, by several orders of magnitude in some cases. As an added bonus, CB also reduced communication overhead in all application scenarios we tested.

Keywords

Disributed Streams; Distributed Monitoring; Resource Limited Devices

1. INTRODUCTION

The following is a canonical problem in distributed systems and databases: given are distributed nodes, and a function which depends on the data at all of them. How can its value be computed, or approximated, with minimal communication? Typically, the trivial solution (collecting all data to a central node and computing the function there) is impractical.

With the advent of *data stream systems* and their increasing importance in quickly evolving fields such as social networks, a more difficult variant of this problem began attracting considerable interest: assume that the data at the nodes is also *dynamic*. Continuously and exactly computing the function's value is typically infeasible, as real-world data consists of many nodes, each holding a large, rapidly changing data stream. This led to the introduction of the *distributed monitoring problem* (also referred to as the *functional monitoring problem*, [29, 7, 34]; see also the survey in [9]), which can be broadly defined as follows:

DEFINITION 1. *Given is a distributed system, with nodes $N_1 ... N_k$, with N_i holding a dynamic data vector $v_i(t)$ (t will be suppressed hereafter to reduce equation clutter). Also given is a function f, which depends on all the v_i's, and a threshold T. The goal is to define* local *conditions at the nodes, such that:*

- Correctness*: As long as all local conditions hold, the global condition $f(v_1 ... v_k) \leq T$ is also guaranteed to hold.*
- Communication efficiency*: The local conditions are "lenient", i.e. the number of times in which they are violated is minimal.*
- Computational efficiency*: The complexity of checking the local conditions is minimal.*

Case study. As a motivating real-life example [26], which applies the Pearson Correlation Coefficient function (treated in this paper), consider a distributed sensor network used to monitor air quality. Often, not only the information on the individual pollutants is important, but the *correlations* between them as well. For example, if an *event* i is defined as pollutant i crossing a certain threshold, one may wish to know whether there exists a correlation between events i, j for two different pollutants. A commonly used such measure, the *Pearson Correlation Coefficient* (PCC), quantifies such a correlation by the value $\frac{z-xy}{\sqrt{x-x^2}\sqrt{y-y^2}}$, where x, y, z are resp. the probabilities of event i, event j, and both events simultaneously. For a distributed system, the global probabilities are averaged over the nodes. It is easy, however, to see that the PCC value of the *global* probabilities can be above a

KDD '16, August 13-17, 2016, San Francisco, CA, USA

DOI: http://dx.doi.org/10.1145/2939672.2939820

given threshold T, while the *local* value at some of the nodes is below T, and vice-versa (for example, in a system with two nodes and local values $x_1 = 0.8, y_1 = 0.2, z_1 = 0.17$ and $x_1 = 0.2, y_1 = 0.7, z_1 = 0.15$, the local PCC values are 0.062 and 0.054, and the global value is -0.26). This is because, for arbitrary functions, there is generally no correlation between the *average of the values* and the *value at the average*.

For general functions, defined over a distributed system, it is typically impossible to determine the position of their global values vis-a-vis T, when given just the local values. The distributed monitoring problem is to impose local conditions which will guarantee that the global value did not cross T.

This problem is known to be rather difficult (NP-complete even in very simple scenarios; see [21]). Nonetheless, considerable progress has been made for real-life problems (Section 2).

Geometric monitoring (GM), introduced in [32], deals with the case in which the monitored function can be expressed as the application of an arbitrary function to the aggregated vector $\frac{v_1+..+v_k}{k}$, i.e. $f = f(\frac{v_1+..+v_k}{k})$. This model turns out to be rich enough to be applicable to a wide range of problems (see Section 2.1). To the best of our knowledge, GM is the only completely generic method for monitoring arbitrary functions over the aggregated data; in this paper, the most recent version of GM [22] is the baseline for comparison.

While a more complete description of GM is deferred to Section 2.1, we note that in order to apply it, the following problem should be repeatedly solved: let S be the hypersurface (or *threshold surface*) defined by $S = \{u | f(u) = T\}$. Then, it is required that each node, at every time-step, calculate the distance of a certain point (unique to that node) from S. This problem can be exceedingly difficult even for surfaces in low-dimensional Euclidean space, and it can render GM unsuitable for monitoring even relatively simple functions, such as the cosine similarity between two vectors (more on this in Section 2.2).

1.1 The convex bound (CB) method

We propose here a very different, simpler, and more direct method to solve the distributed monitoring problem. It relies on the simple observation that, if f is a *convex* function, then, if $f(v_i) \leq T$ holds at every node, it also holds that $f(\frac{v_1+..+v_k}{k}) \leq T$. Thus, monitoring the value of a convex function (from above) is trivial – just monitor its value at every node.

To handle a general f, we propose to search for a convex function c such that $c(u) \geq f(u)$ for all vectors u, and monitor the condition $c \leq T$. This yields a far simpler monitoring condition, whose correctness implies the correctness of the desired condition $f \leq T$. Naturally, the following conditions should hold:

- c should be easy to derive and calculate.
- In order to avoid a high ratio of "false alarms", c should tightly bound f.

We refer to the proposed method as *convex bound* (CB). Clearly, monitoring $f \geq T$ can be similarly achieved by finding a *concave* lower bound.

1.2 Contributions

We offer the following contributions:

- Introducing the CB method and applying it to monitor four popular functions: the Pearson correlation coefficient (PCC hereafter), inner product, cosine similarity, and PCA-Score. These functions were chosen both for their great practical importance and since they do not fall into any category for which there exist simple, efficient solutions (they are not linear, convex, concave, or monotonic).
- Experimentally validating against state-of-the-art methods. CB proved to be far less demanding in terms of local computation at the nodes. Local computation was reduced by one to six orders of magnitude. As an added bonus CB reduced communication overheads for all datasets, functions, and thresholds we tested.
- Proving that every solution GM arrives at can also be obtained with CB.
- Providing a general approach for calculating the convex bound function and proving that it is optimal up to second order Taylor expansion.

2. PREVIOUS WORK

Much of the early work on monitoring distributed streams dealt with the simpler cases of linear functions [20, 19]. Distributed sensor networks were studied in [28, 27]. Other work included top-k monitoring [4]; distributed monitoring of the value of a single-variable polynomial [30], and perturbative analysis of eigenvalues, which was applied to determine local conditions on traffic volume data at the nodes of a distributed system, in order to monitor system health [17]. In [33], the monitoring problem is studied in a probabilistic setting, and in addition to the function's score, a probability threshold is applied; see also [25]. Monitoring entropy was studied in [3]. Ratio queries are handled in [15]. In [35] the norm of the average vector is monitored.

While some problems in monitoring over distributed systems were treated in the past, we are not aware of any general method (capable of handling arbitrary non-linear, non-monotonic, non-convex functions) except for GM and its derivatives, which are surveyed next.

2.1 Previous work on geometric monitoring

In [31, 32] a general approach, *geometric monitoring* (GM), was proposed for tracking the value of a *general* function over distributed streams. GM rests on a geometric result, the so-called *bounding lemma* (details follow in this subsection), which makes it possible to "break up" a global threshold query into conditions that can be checked locally at each site. Followup work [22, 13] proposed various extensions to the basic method. Recent work on GM includes efficient outlier detection in sensor networks [8] and sketch-based monitoring of norm, range-aggregate, and join-aggregate queries over distributed streams [12].

While GM achieved state-of-the-art results in reducing communication overhead for a nice range of central problems, its application is typically hampered by high computational overhead at the nodes. It is this problem which the proposed CB approach aims to alleviate.

Since GM is our baseline for comparison, and it also shares some basic terminology with CB, we next briefly describe it. Proofs and further details can be found in [22] (which is the version that was implemented).

A brief view of GM. Recall that the distributed monitoring problem considers whether $f(\frac{v_1+..+v_k}{k}) \leq T$, where $\{v_i\}$ denote the local dynamic data vectors at the nodes. GM rests on the following geometric interpretation of this question: define the *admissible region*, A, by $A \triangleq \{u|f(u) \leq T\}$. Then, the question is whether the condition $\frac{v_1+..+v_k}{k} \in A$ holds. The first step in answering this question is the following:

LEMMA 1. *[32] Let $v_i(0)$ denote the initial value of the data vector at the i-th node, and let the so-called* reference point*, p_0, be equal to the average of these initial values: $p_0 = \frac{v_1(0)+..+v_k(0)}{k}$. We assume that, during the initial synchronization, a coordinator node broadcasts p_0 to all nodes. Denote the change in the data vector at the i-th node, i.e. $v_i - v_i(0)$, by d_i (it will be referred to as the i-th* drift vector*). Then, the following holds:*

$$v = \frac{v_1 + .. + v_k}{k} = \frac{(p_0 + d_1) + .. + (p_0 + d_k)}{k} \qquad \square$$

Now, the i-th node can independently compute $p_0 + d_i$; and since the global vector v is equal to the average of $p_0 + d_i$, $i = 1..k$, it obviously lies in their convex hull. This can be used to impose *local* conditions on $p_0 + d_i$, which will guarantee that $v \in A$. This is achieved by the *bounding lemma*:

THEOREM 1. *[32] Let B_i denote the (solid) sphere with center $p_0 + d_i/2$ and radius $||d_i||/2$. Then the union $\bigcup_{i=1}^{k} B_i$ contains the convex hull of the vectors $p_0, p_0 + d_1 ... p_0 + d_k$; hence it contains v.*

As a result of the bounding lemma, the local condition used in GM is the following: node i remains silent as long as its sphere B_i is contained in A (Fig. 1). If this condition is violated, the system enters a *violation recovery* phase [32, 12].

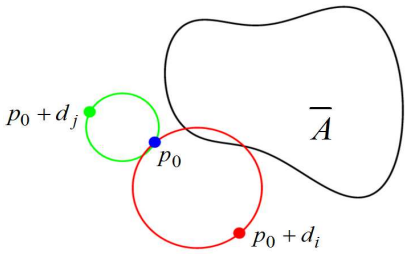


Figure 1: Applying local conditions in GM. The drift vector d_i causes a violation, since the sphere it defines with p_0 intersects the inadmissible region $\bar{A}$; however, d_j does not cause a violation.

2.2 Computational complexity of GM

To apply GM, it must be repeatedly checked whether a certain sphere intersects with $\bar{A}$ – that is, whether its radius is smaller than the distance from its center to A's boundary, which is defined by the *threshold surface* $\{u|f(u) = T\}$. Computing the distance from a point to the threshold surface of a general function is a notoriously difficult problem, and a closed-form solution very rarely exists. Worse, there exist no algorithms which guarantee that the distance will be recovered. Even for the case of a polynomial f, the solution may require an inordinate amount of time; closed-form solutions are often impossible to derive, and iterative schemes are slow and not guaranteed to converge. Known upper bounds on the complexity are extremely high (doubly exponential in the number of variables [2]).

In [24], GM was extended by the *convex decomposition* (CD) approach, which works by decomposing $\bar{A}$ into convex subsets. However, the resulting algorithm has to be specifically tailored to each monitored function, and it suffers from the need to solve the same type of problem as GM (finding the closest point on a surface). For the inner product function, the solution was quite complicated, and we could not apply CD to the PCC or to cosine similarity, which are easily treated by CB.

Clearly, run-times as those often incurred by GM and its derivatives are unacceptable for many distributed streaming systems. We now introduce CB, and demonstrate its advantage for monitoring four popular functions.

3. THE CB METHOD

As noted in the Introduction, it is easy to define local conditions for the monitoring problem (Def. 1) when f is convex: every node i must only check the condition $f(p_0 + d_i) \leq T$ (correctness follows immediately from Lemma 1). We propose to extend this simple observation to monitor arbitrary functions, using an approach which works directly in the realm of functions, as opposed to seeking a geometric solution. The proposed solution works by "relaxing" f to a convex function c that bounds f from above, and monitoring the condition $c \leq T$. This condition both implies $f \leq T$, and is also easy to monitor. We shall refer to c as a *convex bound* for f. Fig. 2 schematically demonstrates the idea behind CB.

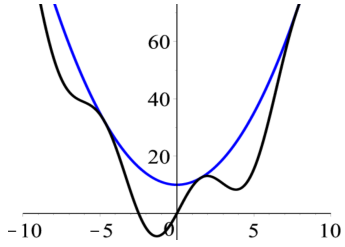


Figure 2: $x^2 + 10$ (blue curve) is a convex bound for $x^2 + 10\sin(x)$ (dark curve).

The next theorem states that every solution which GM provides is also realizable as a solution provided by CB. The proof is omitted due to lack of space.

THEOREM 2. *For every monitoring problem, there is a solution obtained with CB which is exactly identical to the solution obtained with GM – that is, it imposes exactly the same local conditions.*

3.1 Choosing convex bounds

There are, of course, an infinite number of convex bounds

for f, and the question is which of them to choose. To this end, we first propose the following definition.

DEFINITION 2. *Let f be the monitored function. A* tight family *of convex bounds for f, denoted $\mathcal{CB}(f)$, is a set of convex functions satisfying the following requirements:*

- *$g \in \mathcal{CB}(f)$ implies that g is convex and, for every u, $g(u) \geq f(u)$ (the last condition will be denoted $g \succ f$).*
- *Let c be any convex function such that $c \succ f$. Then there exists $g \in \mathcal{CB}(f)$ such that $c \succ g$.*
- *If $g_1, g_2 \in \mathcal{CB}$, then neither $g_1 \succ g_2$ nor $g_2 \succ g_1$.*

Clearly, if g_1, g_2 are both convex bounds for f, and $g_1 \succ g_2$, it is better to use g_2 when monitoring f (since the condition $g_2(v) \leq T$ is weaker than $g_1(v) \leq T$). Therefore we have the following:

LEMMA 2. *When applying CB to monitor f, the convex bound should belong to some family of tight bounds of f.*

In the following case, it is possible to define $\mathcal{CB}(f)$:

LEMMA 3. *Let f be a concave function. Then the family of all tangent planes to f defines a family of tight bounds.*[1]

PROOF. *Every tangent plane is linear, hence convex. Further, it is known that a concave function lies under any of its tangent planes. Now, let g be convex and $g \succ f$. Denote by $U(g)$ the set of all points above g's graph, and by $B(f)$ all points below f's graph. Then both $U(g), B(f)$ are convex, and the minimal distance between them is therefore obtained at points on their boundaries. The tangent plane at the point on f's boundary is the desired element of $\mathcal{CB}(f)$. The idea of the proof is outlined in Fig. 3.* □

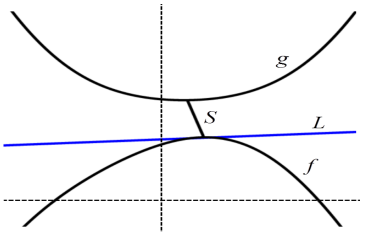


Figure 3: A convex function g and concave function f such that $g \succ f$. S is the segment connecting the two closest points on the graphs. The tangent at the closest point on f's graph, L, satisfies $g \succ L \succ f$, proving that the set of f's tangent planes is a tight family of convex bounds.

3.2 The convexity gap and dependence on the reference point

Replacing the monitored condition $T \geq f$ by $T \geq g \succ f$, for a convex g, enables efficient monitoring – alas, it might also result in potential false alarms (i.e. vectors u for which $T \geq f(u)$ but $T < g(u)$). We refer to this problem as the *convexity gap*, or simply the gap (referring to the gap between f and g). Figuratively speaking, the "system price", one must pay in order to allow distributed monitoring, is reflected in the "convexity price", which is the gap between the monitored function and its convex bound. To minimize the number of false alarms, the gap should be minimized. However, as the following simple example demonstrates, it is often impossible to choose a single optimal g to achieve this goal. As depicted

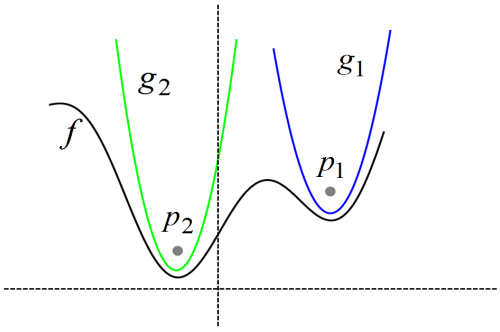


Figure 4: The impossibility of choosing a single best convex bound for the function f (dark curve). g_1 (resp. g_2) is better in the vicinity of p_1 (resp. p_2).

in Fig. 4, it is evident that, loosely speaking, different bounds are better at different regions of the data space, and there is typically no hope of finding a unique bound that is *always* better than all the others. We formalize this observation with the following definition, which is both realizable and appropriate for the monitoring problem:

DEFINITION 3. *A convex bound g_1 is* better than g_2 at a point p *iff there exists a neighborhood of p in which $g_2 \succ g_1$.*

Thus, in Fig. 4, g_i is better at p_i for $i = 1, 2$.

Def. 3 is appropriate for the following reason. Recall that the local condition at the i-th node is $g(p_0 + d_i) \leq T$. Initially, the drift vector d_i is equal to zero; assuming that the data at the nodes behaves continuously, or can be approximated by a random walk ([18, 14, 21, 22]), it follows that the local vector $p_0 + d_i$ can be modeled by a continuous process which starts at p_0 and gradually wanders away from it. Therefore, a bound is sought which is optimal (i.e. smaller than all other bounds) in a certain neighborhood of p_0. For the case of a concave f, such a bound is provided by the following result:

LEMMA 4. *If f is concave, the tangent plane at a point p is the best convex bound at p.*

The proof is trivial, since the tangent plane's value at p is equal to $f(p)$, but all other tangent planes lie above f. Thus the deviation of the tangent plane at p from the point $(p, f(p))$ is quadratic; hence, locally, it is smaller than that of the tangent planes at other points, which is linear. Consequently, when bounding a concave function from above (or, equivalently, a convex function from below), we will replace it with its tangent plane at p_0. This is next used to transform a threshold condition on *general* functions to a convex condition.

3.3 "Convexizing" threshold conditions

If the monitored f is itself convex, the choice of a convex bound c is trivial – choose $c = f$. If f is concave, then,

[1] We deal here with differentiable functions, which include many functions of practical interest. Further, non-differentiable functions can be arbitrarily approximated by differentiable functions on any bounded domain.

following Lemma 4, the tangent plane at p_0 is the optimal candidate for c. We next handle a more general case.

DEFINITION 4. *: Assume that $f = c_1 - c_2$, where both c_1, c_2 are convex. The* convexization *of the condition $f \leq T$ is defined by $c = c_1 - L_{c_2}(p_0) \leq T$, where $L_{c_2}(p_0)$ is the tangent plane of c_2 at p_0.*

Note that the c defined in Def. 4 is convex, bounds f from above, and that its definition is motivated by the special cases where f is convex or concave. The lower bound case is similarly handled: the inequality $f \geq T$ is replaced by the condition $L_{c_1}(p_0) - c_2 \geq T$ (note that $L_{c_1}(p_0) - c_2$ is concave and bounds f from below).

We next prove that, for a very wide class of real problems, it is always possible to express f as the difference of two convex functions. First we recall a definition from calculus that comes in handy for testing convexity:

DEFINITION 5. *Let f be a function of $x_1 ... x_n$. Its* Hessian *H_f is the $n \times n$ matrix $H_f(i,j) = \frac{\partial^2 f}{\partial x_i \partial x_j}$.*

It is well known that a function is convex in a given domain D iff its Hessian is positive semidefinite (PSD) at every point in D[2].

LEMMA 5. *If f possesses bounded second derivatives in a domain D, it can be expressed as the difference of two convex functions.*

PROOF. Since the elements of H_f are bounded over D, there is an upper bound, Λ, on the absolute values of H_f's negative eigenvalues. Define $c_1(u) = f(u) + \frac{\Lambda}{2}||u||^2, c_2(u) = \frac{\Lambda}{2}||u||^2$. Clearly $f = c_1 - c_2$ and c_2 is positive definite. Also, $H_{c_1} = H_f + H_{c_2} = H_f + \Lambda I$ (where I is the identity matrix). Hence all the eigenvalues of H_{c_1} are ≥ 0 and c_1 is convex. □

All the functions we deal with in this paper either have bounded second derivatives, or their derivatives are continuous and the domain of interest is bounded; hence, Lemma 5 is applicable. We will apply it for monitoring cosine similarity (Section 4.3).

The process outlined in Lemma 5 can be extended to provide a convex bound which is optimal to second order Taylor expansion. First, let us formalize the concept of "annihilating" negative eigenvalues:

DEFINITION 6. *Given a symmetric matrix A with spectral decomposition [6] $A = \sum_i \lambda_i e_i e_i^t$ (where λ_i are A's eigenvalues and e_i its eigenvectors), define its* positive part *by $P(A) = \sum_i \max\{\lambda_i, 0\} e_i e_i^t$.*

Theorem 3 (whose proof is omitted due to lack of space) enables to define a convex bound which is optimal to second order.

THEOREM 3. *For a function f and reference point p_0, define a convex bound by $c_{\text{opt}}(p) = f(p_0) + \langle \nabla f(p_0), p - u_0 \rangle + (1/2)(p - p_0) P(H_f(p_0))(p - p_0)^t$. Then, for any other convex bound g of f which satisfies $g(p_0) = f(p_0)$, it holds that $H_g(p_0) \geq H_{c_{\text{opt}}}(p_0)$ (where $\geq$ holds for both the operator and Frobenius norms of the respective Hessians). That is – up to second order, $c_{\text{opt}}(p)$ is an optimal convex bound at the vicinity of p_0.*

[2]A matrix B is PSD iff $uBu^t \geq 0$ for every vector u. A symmetric matrix is PSD iff all its eigenvalues are ≥ 0.

3.3.1 Convexizing inequality constraints

Since $c_1 - c_2 \leq T$ iff $c_1 \leq c_2 + T \triangleq c_3$, we can assume that the monitored condition is given as an inequality between two convex functions, $c_1 \leq c_3$. This condition is especially amenable to convexization: we replace it with $c_1 \leq L_{c_3}(p_0)$, where, as before, $L_{c_3}(p_0)$ is c_3's tangent plane at p_0. We will use this form of convexization for the inner product, cosine similarity, and PCA-Score functions (Section 4).

4. APPLYING CB: THEORY

We now apply CB to monitor four popular functions: Pearson correlation coefficient, inner product, cosine similarity, and PCA-Score ("effective dimension"). In Section 5 we compare the run-time and communication overhead of CB and GM in a variety of real scenarios.

To apply CB, we follow the method described in Section 3.3.1. If the monitored function cannot be directly written as the difference of two convex functions (as in the case of cosine similarity), we apply Lemma 5.

4.1 PCC

Let x, y denote the frequency of appearances of two items in elements of a certain set, and z the frequency of their joint appearances. A very typical example is when x, y denote the ratio of documents in which certain terms appear, and z the same for appearances of both terms simultaneously. The range over which PCC is defined is therefore $0 \leq x, y \leq 1$ and $z \leq x, y$. The function measures the strength of correlation between the appearances of x and y, and is defined by

$$P(x, y, z) = \frac{z - xy}{\sqrt{x - x^2}\sqrt{y - y^2}} \tag{1}$$

We will assume $T > 0$; the case $T \leq 0$ is treated similarly.

The condition $P(x, y, z) \leq T$ can be written as $z \leq xy + T\sqrt{x - x^2}\sqrt{y - y^2}$. We convexize it as follows. First, note that xy is neither convex nor concave; it is simple to verify that the Hessian's eigenvalues for xy are always 1 and -1 (i.e. every point on the function's surface is a *saddle point*). We therefore use the identity $xy = \frac{(x+y)^2}{4} - \frac{(x-y)^2}{4}$. Denote $Q_1 = \frac{(x+y)^2}{4}, Q_2 = \frac{(x-y)^2}{4}$ (note that Q_1, Q_2 are convex). We also need the following:

LEMMA 6. *The function $\sqrt{x - x^2}\sqrt{y - y^2}$ is concave.*

The proof is omitted due to lack of space.

The condition $P(x, y, z) \leq T$ can therefore be written as

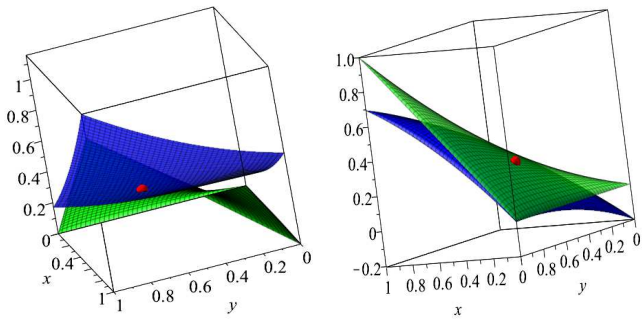
$$(z - T\sqrt{x - x^2}\sqrt{y - y^2} + Q_2) - Q_1 \leq 0 \tag{2}$$

and, since this last expression is the difference of two convex functions[3], we can proceed by applying the paradigm described in Def. 4. The lower bound case is similarly handled. It remains to calculate the tangent planes of $Q_1, Q_2, \sqrt{x - x^2}\sqrt{y - y^2}$, but that is just an exercise in multivariate calculus. The bounds are depicted, for some typical values, in Fig. 5.

4.1.1 Monitoring PCC with GM

As explained in Section 2.2, to apply GM we must be able to solve the closest point problem for the surface defined by $z = xy + T\sqrt{x - x^2}\sqrt{y - y^2}$. To this end we used dedicated software [16], which first reduces the surface's equation to

[3]Since $T\sqrt{x - x^2}\sqrt{y - y^2}$ is concave, its negative is convex.



(a) Upper Bound (b) Lower Bound

Figure 5: Left: a convex upper bound (blue) for PCC (green). The reference point (in red) is $x_0 = 0.3, y_0 = 0.6$, and $T = 0.4$. Right: a concave lower bound.

an algebraic one, and then solves for the closest point. This incurred a run-time far higher than the simple CB solution (by more than three orders of magnitude), and also resulted in higher communication overhead; results are provided in Section 5.2.1.

4.2 Inner product

The inner product function is also extensively applied in data mining and monitoring tasks as a measure of similarity. We assume that the monitored function f is over vectors of length $2n$, and is equal to the inner product of the first and second halves of the vector; denoting the concatenation of vectors x, y by $[x, y]$, we have $f([x, y]) = \langle x, y\rangle$. To express f as the difference of two convex functions, note that $4\langle x, y\rangle = ||x + y||^2 - ||x - y||^2$. Since the norm squared function is convex, the condition $\langle x, y\rangle \leq T$ is convexized by

$$||x + y||^2 \leq 4T + ||x_0 - y_0||^2 + 2\langle [x_0 - y_0, y_0 - x_0], [x - x_0, y - y_0]\rangle \quad (3)$$

where the reference point $p_0 = [x_0, y_0]$, and the gradient of $||x - y||^2$ is equal to $2[x - y, y - x]$ (recall that, for a multivariate function f, the tangent plane at a point u_0 is given by $f(u_0) + \langle \nabla f(u_0), u - u_0\rangle$).

4.2.1 Monitoring inner product with GM

In order to apply GM, one must be able to solve the closest point problem for the threshold surface, $\langle x, y\rangle = T$. If the point outside the surface is denoted $[x_0, y_0]$, the problem can be formulated as

$$\text{Minimize } (||x - x_0||^2 + ||y - y_0||^2) \text{ such that } \langle x, y\rangle = T.$$

This problem can be solved with Lagrange multipliers. Defining $F \triangleq ||x-x_0||^2+||y-y_0||^2+2\lambda(\langle x, y\rangle - T)$ The equations $\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial \lambda} = 0$ assume the form

$$(x - x_0) + \lambda y = 0,\ (y - y_0) + \lambda x = 0,\ \langle x, y\rangle = T \quad (4)$$

These equations can be manipulated to obtain a quartic equation in λ:

$$T\lambda^4 - (2T + \langle x_0, y_0\rangle)\lambda^2 + \left(||x_0||^2 + ||y_0||^2\right)\lambda - \langle x_0, y_0\rangle + T = 0$$

After solving for λ, it is easy to solve for x, y.

While the inner product case is the only one addressed here for which a relatively simple solution for GM could be found, it still incurs the overhead of computing the quartic's coefficients, solving it, and checking the solutions to see which one yields the closest point on the surface. GM's overall run-time was about 20 times higher than CB's.

4.3 Cosine similarity

Another very popular measure of similarity is *cosine similarity* (referred to as *Csim* hereafter), which resembles the inner product function, but normalizes it by the length of the vectors. For example, if we have two histograms of word frequencies, derived from two document corpora, Csim will "neutralize" the effect of the corpus size when measuring the histogram similarity; the inner product function, however, is biased towards larger corpora.

As in the inner product case, the data vector p is $[x, y]$, the concatenation of two n-dimensional vectors x, y, and the reference point will be denoted $p_0 = [x_0, y_0]$. Then, Csim is defined by $\text{Csim}(p) = \frac{\langle x,y\rangle}{||x||||y||}$. Thus, to monitor a lower bound, i.e. $\text{Csim}(p) \geq T$ (we assume $T > 0$; the case of negative T is similarly treated), we need to monitor the condition $\langle x, y\rangle \geq T||x||||y||$. This problem is more complicated than the inner product case, since there is no obvious way to decompose it into an inequality between two convex functions; this is due to the fact that, while representing $\langle x, y\rangle$ as the difference of two convex functions is relatively easy, it is more difficult to derive such a representation for $||x||||y||$. We therefore resort to using the method outlined in Lemma 5. We must first determine the smallest eigenvalue of the Hessian of $||x||||y||$. It follows from the following lemma that it equals -1:

LEMMA 7. *At a point x, y, the eigenvalues of $H(||x||||y||)$ are $1, -1$ (each with multiplicity one) and $||x||/||y||, ||y||/||x||$ (each with multiplicity $n - 1$).*

The proof is omitted due to lack of space. Now we can proceed to convexize the problem. First, we write the inequality $\langle x, y\rangle \geq T||x||||y||$ as $||x+y||^2 \geq ||x-y||^2+4T||x||||y||$. Next, to make both sides convex, we add $2T(||x||^2 + ||y||^2)$ to them, to obtain

$$||x + y||^2 + 2T(||x||^2 + ||y||^2) \geq ||x - y||^2 + 4T||x||||y|| + 2T(||x||^2 + ||y||^2)$$

Lastly, the inequality is convexized by replacing the RHS with its tangent plane at p_0. This step is straightforward, requiring only computation of the gradient, and is omitted for brevity.

4.3.1 Monitoring Csim with GM

The problem of calculating the distance of a point to the Csim surface $\{[x, y]|\langle x, y\rangle = T||x||||y||\}$ is exceedingly difficult. No closed-form solution exists, and three different software packages we applied took about three minutes to complete the task for a *single* point.

4.4 PCA-Score

PCA (Principal Component Analysis) is a fundamental dimension reduction technique with numerous applications. Given a set of vectors in Euclidean space, PCA seeks a low-dimensional subspace which, on the average, well-approximates the vectors in the set. Formally:

DEFINITION 7. *Given $1 > T > 0$ (typically $T \approx 0.9$) and a finite set of vectors $S \subset \mathcal{R}^m$, the* effective dimension *of S*

is defined as the smallest dimension of a sub-space $V \subset \mathcal{R}^m$, *such that* $\sum_{u \in S} ||P_V(u)||^2 \geq T \sum_{u \in S} ||u||^2$, *where* $P_V(u)$ *is the projection of* u *on* V[4].

It is well-known that the effective dimension, denoted k hereafter, can be computed as follows:

1. Construct the $m \times m$ *scatter matrix* $M = \sum_{u \in S} uu^t$. Note that in the distributed setup, S is equal to the sum of local scatter matrices at the nodes.
2. Compute M's eigenvalues, $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_m$.
3. Determining the smallest k such that $\sum_{1 \leq i \leq k} \lambda_i^2 \geq T \sum_{1 \leq i \leq m} \lambda_i^2$.

In [23], PCA was applied to measure the health of a system consisting of distributed nodes. This proceeds as follows: at each timestep, a vector of various system parameters is associated with each node (typically, the vectors' components are various traffic volume indicators). System-wide anomalies (i.e. DDOS attacks) are highly correlated with an increase in the effective dimension of the union of the parameter vectors over all nodes, in a sliding window of pre-determined length.

Hence, the condition to monitor is that the *PCA-Score*, defined by $(\sum_{1 \leq i \leq k} \lambda_i^2)/(\sum_{1 \leq i \leq m} \lambda_i^2)$, is greater than some threshold T. As for the previous functions we handled, the difficulty lies in that λ_i are the eigenvalues of a global matrix which is equal to the sum of the local matrices, hence its exact computation at every timestep will incur a huge communication overhead. In order to apply CB for distributed monitoring, we must express the PCA-Score as a function of the average matrix, as opposed to the sum; however, since (i) eigenvalues scale linearly when the matrix is multiplied by a scalar, and (ii) the PCA-Score is defined as the ratio of sums of squares of eigenvalues, its values on the average and sum matrices are equal.

What remains is to "convexize" the inequality

$$\sum_{1 \leq i \leq k} \lambda_i^2 \geq T \sum_{1 \leq i \leq m} \lambda_i^2. \quad (5)$$

We rely on the following two lemmas:

LEMMA 8. *For an* $m \times m$ *scatter matrix* S, $\sum_{1 \leq i \leq m} \lambda_i^2$ *equals* $\mathrm{Tr}^2(S)$, *and is a convex function of* S. *The proof follows immediately from the fact that every scatter matrix is symmetric.*

LEMMA 9. *For a symmetric* S, $\sum_{1 \leq i \leq k} \lambda_i^2$ *is convex.*

PROOF. *This follows from the famous* Fan identities, *specifically Theorem 2 in [11].* □

Since both sides of Eq. 5 are convex, we can proceed as in Section 3.3.1, by replacing the LHS with the tangent plane at the reference scatter matrix S_0. All that is required is to compute the gradient of the LHS; for that, we use the following result from linear algebra.

[4]We assume that S is *centralized*, i.e. its average is zero. The general case proceeds along the same lines and is omitted due to lack of space.

LEMMA 10. *The derivative of* λ_i *with respect to* S *is equal to* $e_i e_i^t$, *where* e_i *is the eigenvector of* S *corresponding to* λ_i.

Hence, the monitored condition in Eq. 5 is convexized by

$$\sum_{1 \leq i \leq k} \lambda_i^2(S_0) + 2\langle \sum_{1 \leq i \leq k} \lambda_i(S_0) e_i(S_0) e_i^t(S_0), S - S_0 \rangle \geq T(\mathrm{Tr}(S^2)) \quad (6)$$

where S_0 is the reference matrix, and S the local matrix.

4.4.1 Monitoring PCA-Score with GM

In order to apply GM, we must be able to compute the minimal PCA-Score over all matrices in a sphere in the m^2-dimensional space of $m \times m$ matrices. This can be done using *perturbative bounds* that were applied in [17], which also addressed monitoring the health of a distributed system. We also tested a simpler method, analogous to the ones used in [17], in which the safe-zone is defined by the maximal sphere around the reference matrix which is contained in the admissible region. Both methods require bounding the change in the eigenvalues, given the magnitude of change in the matrix. Two such perturbative bounds can be applied, which relate the change in the eigenvalues to the *Frobenius norm* or the *spectral norm* of the change in the matrix. We refer to the algorithms which use the *Frobenius norm* resp. *spectral norm* as FN resp. SN (a detailed description is impossible due to lack of space). We note, however, that all these methods (GM, FN, SN) require solving the difficult problem of finding the closest point on the surface of matrices whose PCA-Score equals T; this renders them slower than CB. Further, CB was better in reducing communication overhead. Details are provided in Section 5.2.4.

5. EXPERIMENTAL EVALUATION

In the experiments, CB was applied to the tasks of monitoring the functions discussed in Section 4, over a few datasets and for different threshold values. For the PCC, Csim and inner product functions we compared CB to GM. To the best of our knowledge, GM represents the state-of-the-art in monitoring threshold queries over distributed streams. We are not aware of any other work on monitoring cosine similarity and the PCC, and while there is other work on monitoring the inner product [10], GM improved on it [12]. For the PCA-score function we compared CB to GM as well as to the Frobenius norm (FN) and spectral norm (SN) perturbative bounds described in [17].

We examined the sliding window scenario, in which the data of interest are the last m records for some pre-defined m (or the last records received within a certain period); for example, one may wish to continuously monitor only the last 1000 tweets in a tweet stream. The sliding window case corresponds to the *turnstile model*, in which the data vector's entries can both increase and decrease, and is more general than the *cash register* model, in which the entries can only increase.

In all the experiments, CB outperformed the other methods in both communication reduction and run-time, with the improvement factor in run-time being orders of magnitude for PCC, cosine similarity, and PCA-Score.

5.1 Data

We used three data sets: the Reuters Corpus (RCV1-v2, REU), a Twitter crawl (Dataset-UDI-TwitterCrawl-Aug2012, TWIT), and the 10 percent sample supplied as part of KDD

Cup 1999 Data (KC). The overall sizes of these data sets were: REU 374MB, TWIT 691MB, KC 46MB.

REU consists of 804,414 news stories, produced by Reuters between August 20, 1996, and August 19, 1997. Each story was categorized according to its content. A total of 47,236 features were extracted from the documents and indexed. Each document is represented as a vector of the features it contains.

TWIT is a subset of Twitter, containing 284 million follower relationships, 3 million user profiles, and 50 million tweets. We filtered the dataset to obtain only hashtagged tweets, which left us with 9 million tweets from 140,000 users. For each tweet, the dataset contains information about the tweet content, ID, creation time, re-tweet count, favorites, hashtags and URLs.

KC was used in the "Third International Knowledge Discovery and Data Mining Tools Competition". It contains information about TCP connections. Each connection is described by 41 features, such as duration, protocol, bytes sent, bytes received etc.

For all data sets, in order to simulate multiple streams, we distributed the data between the nodes in round-robin fashion. Results are presented for 10 streams, and in Section 5.3 we present some results for communication reduction for up to 1,000 streams (the reduction in computational overhead does not depend on the number of streams).

5.2 Computational overhead reduction

Next we summarize the main results of this paper – the reduction in running-time for monitoring the four functions discussed in Section 4. Then we briefly summarize the communication reduction results.

In Fig. 6 we present a summary of the running times for GM and CB, on the various functions and data-sets; details are provided in Sections 5.2.1 to 5.2.4.

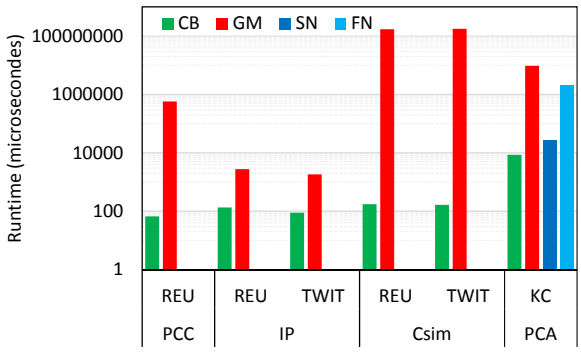


Figure 6: Running times for a local condition check. "SN" and "FN" stand for previous methods to monitor PCA-Score (see Section 5.2.4). Note logarithmic scale.

5.2.1 *Pearson correlation coefficient*

We evaluated PCC on REU, where every document may be labeled as belonging to several categories. The most frequent category is "CCAT" (the "CORPORATE/INDUSTRIAL" category). In the experiments our goal was to select features that are most relevant to this category, i.e. whose PCC with the category is above a given T. Each node holds a sliding window containing the last 6,700 documents it received (this is roughly the number of documents received in a month). We monitored the correlation of "CCAT" with the features "Bosnia" and "Febru".

Run-time evaluation. The majority of GM's run-time is spent on testing for sphere intersection with the PCC surface. To solve this problem we used the Gloptipoly global optimization package [16]. In CB, the local conditions for PCC monitoring are very simple, and only require computing the functions composing the PCC and their derivatives (Section 4.1).

The experiments demonstrated that the run-time of checking the local condition a single time, for the CB method, is almost four orders of magnitude lower than for GM (see Table 1). *Note* – to reduce space, the tables also include results for inner product and Csim[5].

Function	Dim	Run-time (sec.)		Speedup
		GM	CB	
PCC	3	0.58	0.67E-04	8657.7
Inner-Prod	2050	27.4E-04	1.35E-04	20.3
Inner-Prod	1250	18.2E-04	0.89E-04	20.45
Csim	100	170	1.67E-04	1,020,000

Table 1: Run-time for checking the local condition CB vs. GM.

5.2.2 *Inner product*

We monitored the inner product on REU and TWIT. As in [12], we calculated the inner product of feature vectors from two streams (created by splitting the records). For REU, we used the top 2050 features left after removing features which appear in less than 1% of the documents. We used the NLTK [5] package to tokenize and stem the tweets in TWIT, and then selected the top 1250 features, ignoring features appearing in less than 0.1% of the tweets.

In the REU experiment, each node held a sliding window of the last 6,700 documents, while in TWIT each node held a sliding window containing the last 1000 tweets. We used threshold values between 7000 and 17000 for TWIT, and between 4.9E7 to 5.5E7 for REU.

Run-time evaluation. Although GM requires no optimization to find the closest point on the surface, but only to solve a quartic equation, CB checks local conditions about 20 times faster than GM (see Table 1); this is due to the time required to construct and solve the equation, and then check the distinct solutions to see which one yields the closest point. Checking the local conditions requires more time for the REU, since the feature vectors are longer (2050 vs. 1250).

5.2.3 *Cosine similarity*

To evaluate the computational overhead for the cosine similarity function, we monitored both REU and TWIT. Data was the same as for the inner product experiments (see Section 5.2.2 for more details).

Run-time evaluation. The run-time of checking a local condition a single time in GM is almost 3 minutes, while for CB it is less than 0.2 milliseconds (See Table 1).

[5]In the PCA-Score experiments (Section 5.2.4) we compared CB to three different methods, hence the results are provided separately; see Table 2.

5.2.4 PCA-Score

For monitoring the PCA-Score function, we compared CB with GM as well as methods based on the Frobenius norm (FN) and spectral norm (SN) perturbative bounds, described in [17] (see also Section 4.4). All methods except CB require solving complex optimization problems, which were implemented using Matlab and the CVXOPT package [1].

We monitored the PCA-Score over KC using 10 nodes, each holding a sliding window of the last 100 feature vectors. We ran experiments with threshold values T ranging between 0.8 and 0.95, and effective dimension values ranging from 3 to 6.

The experiments show that the three methods which were compared with CB – SN, FN and GM – offer a trade-off between communication cost and run-time.

GM achieves the best communication cost of the three but is also the slowest method. FN is faster than GM but its communication cost is slightly higher. SN is the faster of the three by far, but it achieves relatively poor communication reduction. CB improves on all three methods, achieving better communication cost than GM and better run-time than SN.

Run-time evaluation. Run-time results for monitoring the PCA-Score are displayed in Table 2. The table shows average run-time of a single round of each method the as well as the speedup factor achieved by CB.

CB is about 3 times faster than SN, two orders of magnitude faster than FN, and three orders of magnitude faster than GM. Note that while SN's runtime results are better than GM's, it achieves a rather poor reduction in communication (Fig. 7)

	CB	SN	FN	GM
run-time	0.0086	0.027	2.01	9.57
CB speedup	***1***	***3.20***	***232.81***	***1105.95***

Table 2: Run-times (seconds) for monitoring the PCA-Score over KC.

5.3 Communication overhead reduction

While the work presented here focuses on reducing computational overhead, we also briefly provide results on its performance in reducing overall communication. To evaluate the communication cost, we measured the number of messages sent. The naive method, in which every message is sent to the coordinator, is used as a common baseline. At the opposite extreme, we compared to a hypothetical algorithm, which alerts only when the threshold condition is locally violated, i.e. when $f(v_i) \geq T$ for some local vector v_i. Clearly, every monitoring algorithm will have to alert in such a case. However, to maintain correctness, the local conditions have to adhere to the more restrictive constraint $f(\frac{v_1+\ldots+v_k}{k}) \leq T$. Since the constraints of every correct algorithm are more restrictive, it will issue more alerts, leading to a higher communication cost (unless, of course, f is convex). We refer to this super-optimal bound – the number of local violations – as RLV (real local violations); if the ratio between the number of messages sent by a certain algorithm and the number RLV sent is close to 1, then this algorithm can hardly be improved.

Figure 7 shows a summary of the communication required by CB, GM, and RLV for the functions we studied as well as SN and FN for the PCA-Score function. Each bar represents the results across multiple thresholds and datasets. CB is always better than GM. In most cases CB is close to the super-optimal lower bound RLV, meaning it can be hardly be improved. Note that while FN and SN achieved better runtimes than GM (Table 2) they have higher communication costs. CB did better than the competing methods (GM, FN, SN) in both runtime and communication costs.

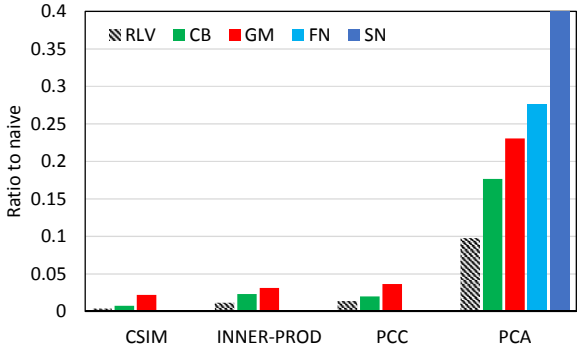


Figure 7: Communication reduction summary (Lower is better). y-axis is ratio to naive. Each bar represents the results across multiple thresholds and datasets. The SN bar is cropped (actual ratio is 1.4)

We also tested the effect of the window size on the communication cost (Fig. 8). The results can be explained by the slower change in the function's value when the window size increases, thus making the monitoring problem easier.

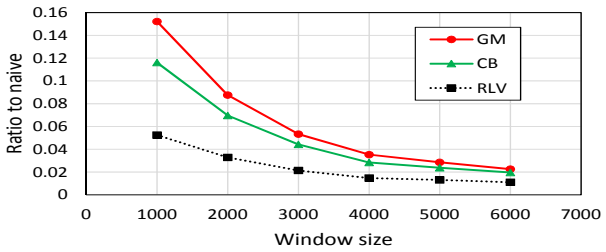


Figure 8: Communication cost as a function of window size for Inner-Prod on TWIT (lower is better).

To test the scalability of CB, we ran experiments with up to 1,000 nodes. Fig. 9 shows the results. The advantage of both CB and GM (and RLV) over the naive grows with the number of nodes, while CB maintains its superiority over GM.

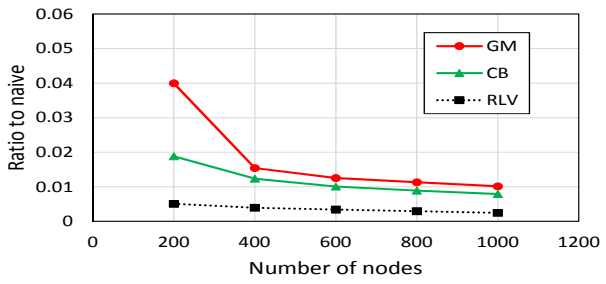


Figure 9: Relative communication cost for up to 1000 nodes (lower is better).

6. CONCLUSIONS

We presented a new method, CB, to monitor threshold functions over distributed streams. The novelty lies in that the monitoring takes place directly over functions, as opposed to previous methods which require solving very difficult optimization problems.

We presented a general paradigm for implementing CB and demonstrated its superiority over previously known methods for four important functions, achieving one to six orders of magnitude run-time improvement, while also reducing the communication cost.

With the move towards the Internet of things, smart-home, smart-cities etc., the deployment of resource-constrained devices is expected to exponentially increase. Systems composed of these devices will have to perform complex monitoring tasks in real-time; hence, the need for computationally efficient solutions, such as the one presented here, is expected to increase.

Future work will concentrate on further applications, as well as on more theoretical directions, e.g. studying alternative methods to "convexize" monitoring problems.

7. ACKNOWLEDGMENTS

The research leading to these results has received funding from the [European Union's] Seventh Framework Programme [FP7-ICT-2013-11] under grant agreement **N**°619491 and **N**°619435 and from the European Commission Horizon 2020-the Framework Programme for Research and Innovation (2014-2020) under grant agreement **N**°688380.

The authors are very grateful to four anonymous reviewers. All their remarks will be incorporated in the planned journal submission.

8. REFERENCES

[1] http://cvxopt.org/.

[2] See survey in http://tinyurl.com/lr4zhrk.

[3] C. Arackaparambil, J. Brody, and A. Chakrabarti. Functional monitoring without monotonicity. In *ICALP (1)*, 2009.

[4] B. Babcock and C. Olston. Distributed top-k monitoring. In *SIGMOD*, New York, NY, USA, 2003. ACM.

[5] S. Bird. Nltk: The natural language toolkit. In *COLING/ACL*, COLING-ACL '06, pages 69–72, 2006.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[7] J. Brody and A. Chakrabarti. A multi-round communication lower bound for gap hamming and some consequences. In *IEEE CCC*, pages 358–368, 2009.

[8] S. Burdakis and A. Deligiannakis. Detecting outliers in sensor networks using the geometric approach. In *ICDE*, 2012.

[9] G. Cormode. The continuous distributed monitoring model. *SIGMOD Record*, 42(1):5–14, 2013.

[10] G. Cormode and M. N. Garofalakis. Approximate continuous querying over distributed streams. *ACM Trans. Database Syst.*, 33(2), 2008.

[11] K. Fan. On a theorem of weyl concerning eigenvalues of linear transformations i. In *Proceedings of the National Academy of Sciences*, volume 35(11), pages 652–655, 1949.

[12] M. N. Garofalakis, D. Keren, and V. Samoladas. Sketch-based geometric monitoring of distributed stream queries. *PVLDB*, 6(10):937–948, 2013.

[13] N. Giatrakos, A. Deligiannakis, M. Garofalakis, I. Sharfman, and A. Schuster. Prediction-based geometric monitoring over distributed data streams. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 265–276. ACM, 2012.

[14] R. Gupta, K. Ramamritham, and M. K. Mohania. Ratio threshold queries over distributed data sources. In *ICDE*, 2010.

[15] R. Gupta, K. Ramamritham, and M. K. Mohania. Ratio threshold queries over distributed data sources. *PVLDB*, 6(8):565–576, 2013.

[16] D. Henrion, J.-B. Lasserre, and J. L.fberg. Gloptipoly 3: moments, optimization and semidefinite programming. *Optimization Methods and Software*, 24(4-5):761–779, 2009.

[17] L. Huang, X. Nguyen, M. N. Garofalakis, J. M. Hellerstein, M. I. Jordan, A. D. Joseph, and N. Taft. Communication-efficient online detection of network-wide anomalies. In *INFOCOM*, 2007.

[18] B. Kanagal and A. Deshpande. Online filtering, smoothing and probabilistic modeling of streaming data. In *ICDE*, 2008.

[19] S. R. Kashyap, J. Ramamirtham, R. Rastogi, and P. Shukla. Efficient constraint monitoring using adaptive thresholds. In *ICDE*, pages 526–535, 2008.

[20] R. Keralapura, G. Cormode, and J. Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In *SIGMOD*, 2006.

[21] D. Keren, G. Sagy, A. Abboud, D. Ben-David, A. Schuster, I. Sharfman, and A. Deligiannakis. Geometric monitoring of heterogeneous streams. *IEEE Trans. Knowl. Data Eng.*, 26(8):1890–1903, 2014.

[22] D. Keren, I. Sharfman, A. Schuster, and A. Livne. Shape sensitive geometric monitoring. *IEEE Trans. Knowl. Data Eng.*, 24(8), 2012.

[23] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM 2004*, pages 219–230, 2004.

[24] A. Lazerson, I. Sharfman, D. Keren, A. Schuster, M. N. Garofalakis, and V. Samoladas. Monitoring distributed streams using convex decompositions. *PVLDB*, 8(5):545–556, 2015.

[25] F. Li, K. Yi, and J. Jestes. Ranking distributed probabilistic data. In *SIGMOD*, 2009.

[26] C. G. O.J. Okunola, A. Uzairu and G. Ndukwe. Assessment of gaseous pollutants along high traffic roads in kano, nigeria. *International Journal of Environment and Sustainability*.

[27] T. Palpanas. Real-time data analytics in sensor networks. In *Managing and Mining Sensor Data*, pages 173–210. 2013.

[28] T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Distributed deviation detection in sensor networks. *SIGMOD Record*, 32(4):77–82, 2003.

[29] J. M. Phillips, E. Verbin, and Q. Zhang. Lower bounds for number-in-hand multiparty communication complexity, made easy. In *SODA*, pages 486–501, 2012.

[30] S. Shah and K. Ramamritham. Handling non-linear polynomial queries over dynamic data. In *ICDE*, 2008.

[31] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. In *SIGMOD*, 2006.

[32] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Trans. Database Syst.*, 32(4), 2007.

[33] M. Tang, F. Li, J. M. Phillips, and J. Jestes. Efficient threshold monitoring for distributed probabilistic data. In *ICDE*, 2012.

[34] R. Wolff. Distributed convex thresholding. In *ACM PODC 2015*, pages 325–334, 2015.

[35] R. Wolff, K. Bhaduri, and H. Kargupta. A generic local algorithm for mining data streams in large distributed systems. *IEEE TKDE*, 21(4), 2009.