Monitoring Distributed Data Streams through Node Clustering

Maria Barouti, Daniel Keren, Jacob Kogan, and Yaakov Malinovsky

University of Maryland Baltimore County, USA and Haifa University, Haifa, Israel
{bmaria2,kogan,yaakovm}@umbc.edu,
dkeren@cs.haifa.ac.il

Abstract. Monitoring data streams in a distributed system is a challenging problem with profound applications. The task of feature selection (e.g., by monitoring the information gain of various features) is an example of an application that requires special techniques to avoid a very high communication overhead when performed using straightforward centralized algorithms.

Motivated by recent contributions based on geometric ideas, we present an alternative approach that combines system theory techniques and clustering. The proposed approach enables monitoring values of an arbitrary threshold function over distributed data streams through a set of constraints applied independently on each stream and/or clusters of streams. The clusters are designed to adapt themselves to the data stream. A correct choice of clusters yields a reduction in communication load. Unlike many clustering algorithms that attempt to collect together similar data items, monitoring requires clusters with *dissimilar* vectors canceling each other as much as possible. In particular, sub–clusters of a good cluster do not have to be good. This novel type of clustering dictated by the problem at hand requires development of new algorithms, and the paper is a step in this direction.

We report experiments on real-world data that detect instances where communication between nodes is required, and show that the clustering approach reduces communication load.

Keywords: data streams, convex analysis, distributed system, clustering.

1 Introduction

In many emerging applications one needs to process a continuous stream of data in real time. Sensor networks [9], network monitoring [6], and real–time analysis of financial data [16], [17] are examples of such applications. Monitoring queries are a particular class of queries in the context of data streams. Previous work in this area deals with monitoring simple aggregates [6], or term frequency occurrence in a set of distributed streams [15]. The current contribution is motivated by results recently reported in [10], [11] where a more general type of monitoring query is described as follows:

Let $\mathbf{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ be a set of data streams collected at n nodes $\mathbf{N} = \{\mathbf{n}_1, \ldots, \mathbf{n}_n\}$. Let $\mathbf{v}_1(t), \ldots, \mathbf{v}_n(t)$ be d-dimensional, real-valued, time varying vectors derived from the streams. For a function $f : \mathbf{R}^d \to \mathbf{R}$ we would like to monitor the inequality

$$f\left(\frac{\mathbf{v}_1(t) + \ldots + \mathbf{v}_n(t)}{n}\right) > 0 \tag{1}$$

while minimizing communication between the nodes. Often the threshold might be a constant r other than 0. In what follows, for notational convenience, we shall always consider the inequality $f > 0$, and when one is interested in monitoring the inequality $f > r$ we will modify the threshold function and consider $g = f - r$, so that the inequality $g > 0$ yields $f > r$. In e.g. [10,7,8,5] a few real-life applications of this monitoring problem are described; see also Section 2 here.

The difference between monitoring problems involving linear and non-linear functions f is discussed and illustrated by a simple example involving a quadratic function f in [10]. The example demonstrates that, for a non-linear f, it is often very difficult to determine from the values of f at the nodes whether its value evaluated at the average vector is above the threshold or not. The present paper deals with the information gain function (see Section 2 for details), and rather than focus on the values of f we consider the location of the vectors $\mathbf{v}_i(t)$ relative to the boundary of the the subset of $\mathbf{R}^d$ where f is positive. We denote this set by $\mathbf{Z}_+(f) = \{\mathbf{v} : f(\mathbf{v}) > 0\}$, and state (1) as

$$\mathbf{v}(t) = \frac{\mathbf{v}_1(t) + \ldots + \mathbf{v}_n(t)}{n} \in \mathbf{Z}_+(f). \tag{2}$$

thus, the functional monitoring problem is transformed to the monitoring of a *geometric* condition. As a simple illustration, consider the case of three scalar functions $v_1(t)$, $v_2(t)$ and $v_3(t)$, and the identity function f (i.e. $f(x) = x$). We would like to monitor the inequality

$$v(t) = \frac{v_1(t) + v_2(t) + v_3(t)}{3} > 0$$

while keeping the nodes silent as long as possible. One strategy is to verify the initial inequality $v(t_0) = \dfrac{v_1(t_0) + v_2(t_0) + v_3(t_0)}{3} > 0$ and to keep the nodes silent while

$$|v_i(t) - v_i(t_0)| < \delta = v(t_0),\ t \geq t_0,\ i = 1, 2, 3.$$

The first time t when one of the functions, say $v_1(t)$, crosses the boundary of the local constraint, i.e. $|v_1(t) - v_1(t_0)| \geq \delta$ the nodes communicate, t_1 is set to be t, the mean $v(t_1)$ is computed, the local constraint δ is updated and made available to the nodes. The nodes are kept silent as long as the inequalities

$$|v_i(t) - v_i(t_1)| < \delta,\ t \geq t_1,\ i = 1, 2, 3$$

hold. This type of monitoring was suggested in [13] for a variety of vector norms. The numerical experiments conducted in [13] with the dataset described in Section 5 show that:

1. The number of time instances the mean violates (1) is a small fraction ($< 1\%$) of the number of time instances when the local constraint is violated at the nodes.
2. The lion's share of communications (about 75%) is required because of a single node violation of the local constraint δ.
3. The smallest number of communications is required when one uses the l_1 norm.

We note that if, for example, the local constraint is violated at $\mathbf{n}_1$, i.e. $|v_1(t) - v_1(t_0)| \geq \delta$, and at the same time

$$v_1(t) - v_1(t_0) = -[v_2(t) - v_2(t_0)],$$

while $|v_3(t) - v_3(t_0)| < \delta$ then $|v(t) - v(t_0)| < \delta$, $f(v(t)) > 0$, and update of the mean can be avoided. Separate monitoring of the two node cluster $\{\mathbf{n}_1, \mathbf{n}_2\}$ would require communication involving two nodes only, and could reduce communication load. We aim to extend this idea to the general case – involving many nodes, arbitrary functions, and high-dimensional data.

Clustering in general is a difficult problem, and many clustering problems are known to be NP-complete [4]. Unlike standard clustering that attempts to collect together similar data items [2], we are seeking clusters with *dissimilar data items*, which cancel out each other as much as possible. While sub-clusters of a "classical" good cluster are usually good, this may not be the case when a cluster contains dissimilar objects. These observations indicate that common clustering methods are not applicable to our problem.

A basic attempt to cluster nodes was suggested in [14] with results reported for the dataset presented in Section 5. Clustering together just two nodes reported in [14] reduces communication by about 10%.

In this paper we advance clustering approach to monitoring. The main contribution of this work is twofold:

1. We suggest a specific clustering strategy, and report the communication reduction achieved.
2. We apply the same clustering strategy with l_1, l_2, and l_∞ norms and report the results obtained.

The paper is organized as follows. In Section 2 we present a relevant Text Mining application. Section 3 provides motivation for node clustering. A specific implementation of node clustering is presented in Section 4. Experimental results are reported in Section 5. Section 6 concludes the paper and indicates new research directions.

In the next section we provide a Text Mining related example that leads to a non linear threshold function f.

2 Text Mining Application

Let $\mathbf{T}$ be a textual database (for example a collection of mail or news items). We denote the size of the set $\mathbf{T}$ by $|\mathbf{T}|$. We follow the methodology suggested in [10] and assume that some of the documents are marked as "spam," and a "feature" (word or term for example) is selected. We will be concerned with two subsets of $\mathbf{T}$:

1. $\mathbf{R}$–the set of "relevant" texts (e.g. texts not labeled as "spam"),
2. $\mathbf{F}$–the set of texts that contain a "feature."

We denote complements of the sets by $\overline{\mathbf{R}}$, $\overline{\mathbf{F}}$ respectively (i.e. $\mathbf{R}\cup\overline{\mathbf{R}} = \mathbf{F}\cup\overline{\mathbf{F}} = \mathbf{T}$), and consider the relative size of the four sets $\mathbf{F}\cap\overline{\mathbf{R}}$, $\mathbf{F}\cap\mathbf{R}$, $\overline{\mathbf{F}}\cap\overline{\mathbf{R}}$, and $\overline{\mathbf{F}}\cap\mathbf{R}$ associated with text collection and the "feature" as follows:

$$\begin{aligned} x_{11}(\mathbf{T}) &= \frac{|\mathbf{F}\cap\overline{\mathbf{R}}|}{|\mathbf{T}|}, \; x_{12}(\mathbf{T}) = \frac{|\mathbf{F}\cap\mathbf{R}|}{|\mathbf{T}|}, \\ x_{21}(\mathbf{T}) &= \frac{|\overline{\mathbf{F}}\cap\overline{\mathbf{R}}|}{|\mathbf{T}|}, \; x_{22}(\mathbf{T}) = \frac{|\overline{\mathbf{F}}\cap\mathbf{R}|}{|\mathbf{T}|}. \end{aligned} \tag{3}$$

Note that the non negative numbers x_{ij} depend on the "feature",

$$0 \le x_{ij} \le 1, \text{ and } x_{11} + x_{12} + x_{21} + x_{22} = 1.$$

The function f is defined on the simplex (i.e. $x_{ij} \ge 0$, $\sum x_{ij} = 1$), and given by

$$\sum_{i,j} x_{ij} \log\left(\frac{x_{ij}}{(x_{i1} + x_{i2})(x_{1j} + x_{2j})}\right), \tag{4}$$

where $\log x = \log_2 x$ throughout the paper. It is well-known that (4) provides the *information gain* for the "feature" (see e.g. [1]).

As an example, we consider n agents installed on n different servers, and a stream of texts arriving at the servers. Let $\mathbf{T}_h = \{\mathbf{t}_{h1}, \dots, \mathbf{t}_{hw}\}$ be the last w texts received at the h^{th} server, with $\mathbf{T} = \bigcup_{h=1}^{n} \mathbf{T}_h$. Note that

$$x_{ij}(\mathbf{T}) = \sum_{h=1}^{n} \frac{|\mathbf{T}_h|}{|\mathbf{T}|} x_{ij}(\mathbf{T}_h),$$

i.e., entries of the global contingency table $\{x_{ij}(\mathbf{T})\}$ are the weighted average of the local contingency tables $\{x_{ij}(\mathbf{T}_h)\}$, $h = 1, \dots, n$.

To check that the given "feature" is sufficiently informative with respect to the target relevance label r, one may want to monitor the inequality

$$f\left(x_{11}(\mathbf{T}), x_{12}(\mathbf{T}), x_{21}(\mathbf{T}), x_{22}(\mathbf{T})\right) - r > 0 \tag{5}$$

with f given by (4) while minimizing communication between the servers.

In the next section we provide motivation to node clustering for monitoring data streams.

3 Monitoring Threshold Functions through Clustering: Motivation

In what follows we denote a norm of a vector $\mathbf{v}$ by $\|\mathbf{v}\|$. While the experiments reported in this paper have been conducted with l_1, l_2, and l_∞ norms, the proposed monitoring and node clustering procedures can be applied with any norm. The monitoring strategy proposed in [13] can be briefly described as follows:

Algorithm 31 *Monitoring Threshold Function*

```
– A node is designated as a root $\mathbf{r}$.
– The root sets $i = 0$.
– Until end of stream
    1. The root sends a request to each node $\mathbf{n}$ for the vectors $\mathbf{v_n}(t_i)$. The nodes
       respond to the root. The root computes the distance $\delta$ between the mean
       $\frac{1}{n}\sum_{\mathbf{n}\in\mathbf{N}} \mathbf{v}_n(t_i)$ and the zero set $\mathbf{Z}_f$ of the function $f$. The root transmits
       $\delta$ to each node.
    2. do for each $\mathbf{n} \in \mathbf{N}$
       If $||\mathbf{v_n}(t) - \mathbf{v_n}(t_i)|| < \delta$
            the node $\mathbf{n}$ is silent
       else
            $\mathbf{n}$ notifies the root about violation of its local constraint $\delta$
            the root sets $i = i + 1$
            go to Step 1.
– Stop
```

An application of the above procedure to data streams generated from the Reuters Corpus RCV1–V2 (see Section 5 for detailed description of the data and experiments) leads to 4006 time instances in which the local constraints are violated, and the root is updated. Results presented in Table 1 show that in 3034 out of 4006 time instances, communications with the root are triggered by constraint violations at exactly one node.

The results immediately suggest to cluster nodes to further reduce communication load. Indeed clustering together, for example, the "longest" vector $\mathbf{v}_{\mathbf{n}_L}(t) - \mathbf{v}_{\mathbf{n}_L}(t_i)$ with the "shortest" vector $\mathbf{v}_{\mathbf{n}_S}(t) - \mathbf{v}_{\mathbf{n}_S}(t_i)$ may result in the mean of the two vector cluster being shorter than δ. In such a case communication involving only two rather than all n nodes may prevent mean update for the entire set of nodes in many of the 3034 instances listed in Table 1. Clustering together just two nodes as described above reported in [14] reduces communication by about 10%.

Table 1. Number of local constraint violations simultaneously by k nodes, $r = 0.0025$, l_2 norm, the feature is "bosnia"

# of nodes violators	1	2	3	4	5	6	7	8	9	10
# of violation instances	3034	620	162	70	38	26	34	17	5	0

In this paper we advance the node clustering approach and demonstrate additional communication savings. Each cluster will be equipped with a "coordinator" $\mathbf{c}$ (one of the cluster's nodes). If a cluster node $\mathbf{n}$ violates its local constraint at time t, then the coordinator collects vectors $\mathbf{v}_{\mathbf{n}}(t) - \mathbf{v}_{\mathbf{n}}(t_i)$ from all the nodes in the cluster, computes the mean of the vectors, and checks whether the mean violates the coordinator constraint δ (at this point, node and coordinator constraints are identical). We shall follow [10] and refer to this step as "the balancing process." If the coordinator constraint is violated, the coordinator alerts the root, and the mean of the entire dataset is recomputed by the root (for detailed description of the procedure see Section 4).

A standard clustering problem is often described as "...finding and describing cohesive or homogeneous chunks in data, the clusters" (see e.g. [2]). For the problem at hand we would like to partition the set of nodes $\mathbf{N}$ into k clusters $\Pi = \{\pi_1, \ldots, \pi_k\}$ so that

$$\mathbf{N} = \bigcup_{i=1}^{k} \pi_i, \text{ and } \pi_i \bigcap \pi_j = \emptyset \text{ if } i \neq j.$$

We denote the size of π_i by $|\pi_i|$. If for each cluster π_i one has

$$\frac{1}{|\pi_i|} \left\| \sum_{\mathbf{n} \in \pi_i} [\mathbf{v}_{\mathbf{n}}(t) - \mathbf{v}_{\mathbf{n}}(t_j)] \right\| < \delta, \tag{6}$$

then, due to convexity of any norm, one has

$$\left\| \frac{1}{n} \sum_{\mathbf{n} \in \mathbf{N}} \mathbf{v}_{\mathbf{n}}(t) - \frac{1}{n} \sum_{\mathbf{n} \in \mathbf{N}} \mathbf{v}_{\mathbf{n}}(t_j) \right\| \leq \sum_{i=1}^{k} \frac{|\pi_i|}{n} \left[\frac{1}{|\pi_i|} \left\| \sum_{\mathbf{n} \in \pi_i} [\mathbf{v}_{\mathbf{n}}(t) - \mathbf{v}_{\mathbf{n}}(t_j)] \right\| \right] < \delta.$$

The inequality shows that the "new" mean $\frac{1}{n} \sum_{\mathbf{n} \in \mathbf{N}} \mathbf{v}_{\mathbf{n}}(t)$ belongs to $\mathbf{Z}_+(f)$ if the distance from the "old" mean $\frac{1}{n} \sum_{\mathbf{n} \in \mathbf{N}} \mathbf{v}_{\mathbf{n}}(t_j)$ to the boundary of this set exceeds δ, and (6) holds for each cluster. We therefore may attempt to define the quality of a k cluster partition Π as

$$Q(\Pi) = \max_i \left\{ \frac{1}{|\pi_i|} \left\| \sum_{\mathbf{n} \in \pi_i} [\mathbf{v}_{\mathbf{n}}(t) - \mathbf{v}_{\mathbf{n}}(t_j)] \right\|, \; i = 1, \ldots, k \right\}. \tag{7}$$

Our aim is to identify k and a k cluster partition Π^o that **minimizes** (7). Our monitoring problem requires to assign nodes $\{\mathbf{n}_{i_1}, \ldots, \mathbf{n}_{i_l}\}$ to the same cluster π so that the total average change within cluster

$$\left\| \frac{1}{|\pi|} \sum_{\mathbf{n} \in \pi} [\mathbf{v}_{\mathbf{n}}(t) - \mathbf{v}_{\mathbf{n}}(t_j)] \right\| \text{ for } t > t_j$$

is minimized, i.e., nodes with **different** variations $\mathbf{v_n}(t)-\mathbf{v_n}(t_j)$ that cancel out each other as much as possible are assigned to the same cluster. Hence, unlike classical clustering procedures, this one needs to combine "dissimilar" nodes together.

The proposed partition quality $Q(\Pi)$ (see (7)) generates three immediate problems:

1. Since the arithmetic mean $\overline{a}$ of a finite set of real numbers $\{a_1, \ldots, a_k\}$ satisfies
$$\min\{a_1, \ldots, a_k\} \leq \overline{a} \leq \max\{a_1, \ldots, a_k\}$$
the single cluster partition always minimizes $Q(\Pi)$. Considering the entire set of nodes as a single cluster with its own coordinator that communicates with the root introduces an additional unnecessary "bureaucracy" layer that only increases communications. We seek a trade-off which yields clusters with "good" sizes (this is rigorously defined in Section 4).
2. Computation of $Q(\Pi)$ involves future values $\mathbf{v_n}(t)$, which are not available at time t_j when the clustering is performed.
3. Since the communication overhead of the balancing process is proportional to the size of a cluster, the individual clusters' sizes should affect the clustering quality $q(\pi)$ (see (8) below).

In the next section we address these problems.

4 Monitoring Threshold Functions through Clustering: Implementation

We argue that in addition to the average magnitude of the variations $\mathbf{v_n}(t) - \mathbf{v_n}(t_j)$ inside the cluster π, the cluster's size also affects the frequency of updates, and, as a result, the communication load. We therefore define the quality of the cluster π by
$$q(\pi) = \frac{1}{|\pi|}\left|\left|\sum_{\mathbf{n}\in\pi}[\mathbf{v_n}(t) - \mathbf{v_n}(t_j)]\right|\right| + \alpha|\pi|, \tag{8}$$
where α is a nonnegative scalar parameter. The quality of the partition $\Pi = \{\pi_1, \ldots, \pi_k\}$ is defined by
$$Q(\Pi) = \max_{i\in\{1,\ldots,k\}} q(\pi_i), \tag{9}$$

When $\alpha = 0$ the partition that minimizes $Q(\Pi)$ is a single cluster partition (that we would like to avoid). When $\max_{\mathbf{n}} ||\mathbf{v_n}(t) - \mathbf{v_n}(t_j)|| \leq \alpha$ the optimal partition is made up of n singleton clusters. In this paper we focus on
$$0 < \alpha < \max_{\mathbf{n}\in\mathbf{N}} ||\mathbf{v_n}(t) - \mathbf{v_n}(t_j)||\,. \tag{10}$$

The constant α depends on t and t_j, and below we show how to avoid this dependence.

Computation of $Q(\Pi)$ required for the clustering procedure is described below. In order to compute $Q(\Pi)$ at time t_j one needs to know $\mathbf{v}_{\mathbf{n}}(t)$ at future times $t = t_j + 1, t_j + 2, \ldots$ which are not available (we recall that t_j denotes time instances when the mean of the entire data set is updated). While the future behavior is not known, we shall use past values of $\mathbf{v}_{\mathbf{n}}(t)$ for prediction. For each node $\mathbf{n}$ we build "history" vectors $\mathbf{h}_{\mathbf{n}}(t_j)$ defined as follows:

```
1. $\mathbf{h}_{\mathbf{n}}(t_0) = 0$
2. if ($\mathbf{h}_{\mathbf{n}}(t_j)$ is already available)
        $\mathbf{h}_{\mathbf{n}}(t_{j+1}) = \mathbf{h}_{\mathbf{n}}(t_j)$
        for $t$ increasing from $t_j$ to $t_{j+1}$ do
            $\mathbf{h}_{\mathbf{n}}(t_{j+1}) = \frac{1}{2}\mathbf{h}_{\mathbf{n}}(t_{j+1}) + [\mathbf{v}_{\mathbf{n}}(t) - \mathbf{v}_{\mathbf{n}}(t_j)]$
```

The vectors $\mathbf{h}_{\mathbf{n}}(t_j)$ accumulate the history of changes, with older changes assigned smaller weights. In this paper we arbitrary use $\frac{1}{2}$ as the the weight. It is clear that other values can be used as weights, furthermore the weights do not have to be constants, and may reflect known history of changes.

We shall use the vectors $\{\mathbf{h}_{\mathbf{n}}(t_j)\}$ to generate a node partition at time t_j. We note that normalization of the vector set that should be clustered does not change the induced optimal partitioning of the nodes. When the vector set is normalized by the magnitude of the longest vector in the set, the range for α conveniently shrinks to $[0, 1]$. In what follows we set $h = \max_{\mathbf{n}} ||\mathbf{h}_{\mathbf{n}}(t_j)||$, assume that $h > 0$, and describe a "greedy" clustering procedure for the normalized vector set

$$\{\mathbf{a}_1, \ldots, \mathbf{a}_n\},\ \mathbf{a}_i = \frac{1}{h}\mathbf{h}_{\mathbf{n}_i}(t_j),\ i = 1, \ldots, n.$$

We start with the n cluster partition Π^n (each cluster is a singleton). If a k cluster partition Π^k, $k > 2$ is already available we

1. identify the partition cluster π_j with maximal norm of its vectors' mean, i.e.,

$$\frac{1}{|\pi_j|}\left|\left|\sum_{\mathbf{a}\in\pi_j}\mathbf{a}\right|\right| \geq \frac{1}{|\pi_i|}\left|\left|\sum_{\mathbf{a}\in\pi_i}\mathbf{a}\right|\right|,\ i = 1, \ldots, n.$$

2. identify cluster π_i so that the merger of π_i with π_j produces a cluster of smallest possible quality, i.e.,

$$q\left(\pi_j \bigcup \pi_i\right) \leq q\left(\pi_j \bigcup \pi_l\right),\ l \neq j,$$

   where cluster's quality is defined by (8).

The partition Π^{k-1} is obtained from Π^k by merging clusters π_j and π_i. The final partition is selected from the $n-1$ partitions $\{\Pi^2, \ldots, \Pi^n\}$ as the one that minimizes Q.

Note that node constraints δ do not have to be equal. Taking into account the distribution of the data streams at each node can further reduce communication.

We illustrate this statement by a simple example involving two nodes. If, for example, there is reason to believe that the inequality

$$2\|\mathbf{v}_1(t) - \mathbf{v}_1(t_i)\| \le \|\mathbf{v}_2(t) - \mathbf{v}_2(t_i)\| \tag{11}$$

always holds, then the number of node violations may be reduced by imposing node dependent constraints

$$\|\mathbf{v}_1(t) - \mathbf{v}_1(t_i)\| < \delta_1 = \frac{2}{3}\delta, \text{ and } \|\mathbf{v}_2(t) - \mathbf{v}_2(t_i)\| < \delta_2 = \frac{4}{3}\delta$$

so that the wider varying stream at the second node enjoys larger “freedom” of change, while the inequality

$$\left\|\frac{\mathbf{v}_1(t) + \mathbf{v}_2(t)}{2} - \frac{\mathbf{v}_1(t_i) + \mathbf{v}_2(t_i)}{2}\right\| < \frac{\delta_1 + \delta_2}{2} = \delta$$

holds true. Assigning “weighted” local constraints requires information provided by (11). With no additional assumptions about the stream data distribution this information is not available. Unlike [12] we refrain from making assumptions regarding the underlying data distributions; instead, we estimate the weights through past values $\|\mathbf{v}_j(t) - \mathbf{v}_j(t_i)\|$.

1. Start with the initial set of weights

$$w_1 = \ldots = w_n = 1 \text{ and } W_1 = \ldots = W_n = 1 \tag{12}$$

(so that $\sum_{j=1}^{n} w_j = \sum_{j=1}^{n} W_j = n$).

2. As new texts arrive at the next time instance t, each node computes updates

$$W_j = \frac{1}{2}W_j + \|\mathbf{v}_j(t) - \mathbf{v}_j(t_i)\|, \text{ with } W_j(t_0) = 1, \; j = 1, \ldots, n.$$

When at time t_{i+1} the root constraint $\delta(\mathbf{r})$ needs to be updated, each node $\mathbf{n}_j$ broadcasts W_j to the root. The root computes $W = \sum_{j=1}^{n} W_j$, and transmits the updated $\delta(\mathbf{n}_j) = w_j\delta(\mathbf{r})$ where $w_j = n \times \frac{W_j}{W}$ (so that $\sum_{j=1}^{n} w_j = n$) back to node j. For a coordinator $\mathbf{c}$ of a node cluster π the constraint $\delta(\mathbf{c}) = \frac{1}{|\pi|}\sum_{\mathbf{n}\in\pi}\delta(\mathbf{n})$.

5 Experimental Results

The data streams analyzed in this section are generated from the Reuters Corpus RCV1–V2. The data is available from `http://leon.bottou.org/projects/sgd` and consists of 781, 265 tokenized documents with document ID ranging from 2651 to 810596. We simulate n streams by arranging the feature vectors in ascending order with respect to document ID, and selecting feature vectors for the stream in the round-robin fashion.

In the Reuters Corpus RCV1–V2 each document is labeled as belonging to one or more categories. We label a vector as "relevant" if it belongs to the "CORPORATE/INDUSTRIAL" ("CCAT") category, and "spam" otherwise. Following [10] we focus on three features: "bosnia," "ipo," and "febru." Each experiment was performed with 10 nodes, where each node holds a sliding window containing the last 6,700 documents it received.

First we use 67, 000 documents to generate initial sliding windows. The remaining 714, 265 documents are used to generate datastreams, hence the selected feature information gain is computed 714, 265 times. Based on all the documents contained in the sliding window at each one of the 714, 266 time instances we compute and graph 714, 266 information gain values for the feature "bosnia" (see Figure 1).

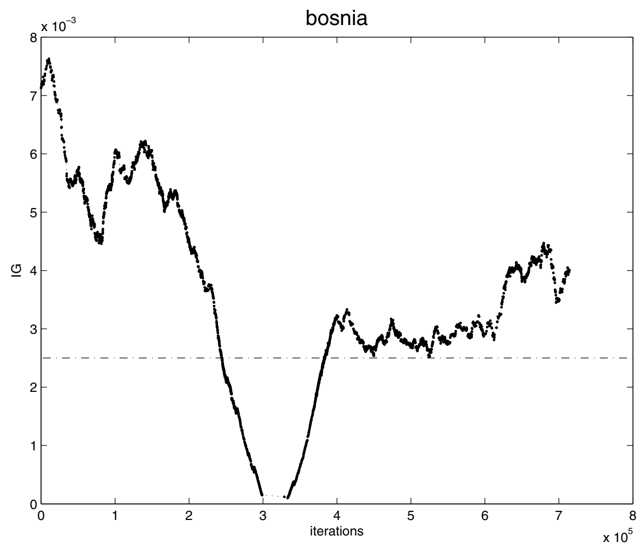

Fig. 1. Information gain values for the feature "bosnia"

For the experiments described below, the threshold value r is predefined, and the goal is to monitor the inequality $f(\mathbf{v}) - r > 0$ while minimizing communication between the nodes.

We assume that new texts arrive simultaneously at each node. The numerical experiment reported in [13] with the feature "febru," and the threshold $r = 0.0025$ are shown in Table 2 where a broadcast is defined as one time transmission of information between different nodes. We run the node clustering monitoring presented in this paper for the same feature and threshold with

Table 2. Number of mean computations, and broadcasts for feature "febru" with threshold $r = 0.0025$, no clustering

norm	mean updates	broadcasts
l_1	2591	67388
l_2	3140	81650
l_∞	3044	79144

Table 3. Number of root and coordinator mean computations, and total broadcasts for feature "febru" with threshold $r = 0.0025$ with clustering

norm	alpha	root mean update	coordinator mean update	total broadcasts
l_1	0.70	1431	0	38665
l_2	0.80	1317	0	35597
l_∞	0.65	1409	0	38093

$\alpha = 0.05, 0.10, \ldots, 0.95$. The best results for l_1, l_2, and l_∞ norms with respect to α are presented in Table 3. The clustering approach in this case is particularly successful – coordinators' constraints are not violated, and the root mean updates are decreased significantly. As a result the number of broadcasts decreases by about 50%.

Next we turn to the features "ipo" and "bosnia." In both cases we run monitoring with clustering, allowing $\alpha = 0.05, 0.10, \ldots, 0.95$, and report results with the lowest number of broadcasts. The results obtained for "ipo" without clustering are presented in Table 4. Application of clustering procedure leads to a significant reduction in the number of broadcasts. Results obtained through clustering procedure are shown in Table 5. The table demonstrates significant inside cluster activity, and a decrease in root mean updates.

Table 4. Number of mean computations, and broadcasts for feature "ipo" with threshold $r = 0.0025$, no clustering

norm	mean updates	broadcasts
l_1	15331	398606
l_2	21109	548834
l_∞	19598	509548

Finally we turn to the feature "bosnia." Application of clustering to monitoring this feature information gain appears to be far less successful. Results obtained without clustering in [13] are presented in Table 6. Application of the clustering procedure leads to a slight decrease in the number of broadcasts in case of the l_2 and l_∞ norms (see Table 7). In case of the l_1 norm, the number of broadcasts increases. Clustering does not offer a universal remedy; in some

Table 5. Number of root and coordinator mean computations, and total broadcasts for feature "ipo" with threshold $r = 0.0025$ with clustering

norm	alpha	root mean update	coordinator mean update	total broadcasts
l_1	0.15	5455	829	217925
l_2	0.10	7414	1782	296276
l_∞	0.10	9768	2346	366300

Table 6. Number of mean computations, and broadcasts, for feature "bosnia" with threshold $r = 0.0025$, no clustering

norm	mean updates	broadcasts
l_1	3053	79378
l_2	4006	104156
l_∞	3801	98826

Table 7. Number of root and coordinator mean computations, and total broadcasts for feature "bosnia" with threshold $r = 0.0025$ and clustering

norm	alpha	root mean update	coordinator mean update	total broadcasts
l_1	0.65	3290	2	89128
l_2	0.55	3502	7	97602
l_∞	0.60	3338	2	91306

cases better performance is achieved with no clustering (by keeping α between 0.05 and 0.95 we force nodes to cluster).

6 Conclusions and Future Research Directions

In this paper we propose to monitor threshold functions over distributed data streams through clustering nodes whose data fluctuations "cancel out" each other. The strategy, if successful, in many cases reduces the communication required to message exchanges within a cluster only, yielding overall communication reduction.

The clustering strategy suggested is based on minimization of a combination of average of a vector associated with a cluster and the cluster size. The nodes are re–clustered each time the entire dataset mean violates its constraint $\delta(\mathbf{r})$. The amount of communication required depends on the "trade off" parameter $0 < \alpha < 1$ selected at the beginning of the monitoring process. While the results obtained show improvement over previously reported ones that do not use clustering [13] it is of interest to introduce an update of α based on the monitoring history each time nodes are re–clustered (see e.g. [3] for feedback theory approach).

Clustering does not provide a universal remedy. It is of interest to identify data streams that benefit from clustering, and those for which clustering does not reduce communication load in any significant fashion. Finally a methodology that measures effectiveness of various monitoring techniques should be introduced, so that different monitoring strategies can be easily compared.

Acknowledgment. The authors thank the reviewers for the valuable remarks that much improved exposition of the results. The research of the second author was supported by Grant No. 2008405 from the United States-Israel Binational Science Foundation (BSF). The work of the fourth author was partially supported by a 2013 UMBC Summer Faculty Fellowship grant.

References

1. Gray, R.M.: Entropy and Information Theory. Springer, New York (1990)
2. Mirkin, B.: Clustering for Data Mining: A Data Recovery Approach. Chapman & Hall/CRC, Boca Raton (2005)
3. Willems, J.C.: The Analysis of Feedback Systems. The MIT Press, Cambridge (1971)
4. Brucker, P.: On the complexity of clustering problems. Lecture Notes in Economics and Mathematical Systems, vol. 157, pp. 45–54 (1978)
5. Burdakis, S., Deligiannakis, A.: Detecting outliers in sensor networks using the geometric approach. In: ICDE, pp. 1108–1119 (2012)
6. Dilman, M., Raz, D.: Efficient reactive monitoring. In: Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communication Societies, pp. 1012–1019 (2001)
7. Gabel, M., Schuster, A., Keren, D.: Communication-efficient outlier detection for scale-out systems. In: BD3@VLDB, pp. 19–24 (2013)
8. Garofalakis, M., Keren, D., Samoladas, V.: Sketch-based geometric monitoring of distributed stream queries. In: PVLDB (2013)
9. Madden, S., Franklin, M.J.: An architecture for queries over streaming sensor data. In: ICDE 2002, p. 555 (2002)
10. Sharfman, I., Schuster, A., Keren, D.: A Geometric Approach to Monitoring Threshold Functions over Distributed Data Streams. ACM Transactions on Database Systems 32, 23:1–23:29 (2007)
11. Sharfman, I., Schuster, A., Keren, D.: A Geometric Approach to Monitoring Threshold Functions over Distributed Data Streams. In: May, M., Saitta, L. (eds.) Ubiquitous Knowledge Discovery. LNCS, vol. 6202, pp. 163–186. Springer, Heidelberg (2010)
12. Keren, D., Sharfman, I., Schuster, A., Livne, A.: Shape Sensitive Geometric Monitoring. IEEE Transactions on Knowledge and Data Engineering 24, 1520–1535 (2012)
13. Kogan, J.: Feature Selection over Distributed Data Streams through Convex Optimization. In: Proceedings of the Twelfth SIAM International Conference on Data Mining (SDM 2012), pp. 475–484. SIAM (2012)

14. Kogan, J., Malinovsky, Y.: Monitoring Threshold Functions over Distributed Data Streams with Clustering. In: Proceedings of the Workshop on Data Mining for Service and Maintenance (held in conjunction with the 2013 SIAM International Conference on Data Mining), pp. 5–13 (2013)
15. Manjhi, A., Shkapenyuk, V., Dhamdhere, K., Olston, C.: Finding (recently) frequent items in distributed data streams. In: ICDE 2005, pp. 767–778 (2005)
16. Yi, B.-K., Sidiropoulos, N., Johnson, T., Jagadish, H.V., Faloutsos, C., Biliris, A.: Online datamining for co–evolving time sequences. In: ICDE 2000 (2000)
17. Zhu, Y., Shasha, D.: Statestream: Statistical monitoring of thousands of data streamsin real time. In: VLDB, pp. 358–369 (2002)