Monitoring Distributed Data Streams Through Node Clustering

Maria Barouti[1], Daniel Keren[2], Jacob Kogan[3], Yaakov Malinovsky[4]

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 addressed 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 adopt themselves to the data stream. A correct choice of clusters yields a reduction in communication load. We report experiments on a real–world data that detect instances where communication between nodes is required, and show that the clustering approach reduces communication load.

Keyword list: 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 [7], network monitoring [2], and real–time analysis of financial data [13], [14] are examples of such applications. Monitoring queries is a particular class of queries in the context of data streams. Previous work in this area deals with monitoring simple aggregates [2], or term frequency occurrence in a set of distributed streams [8]. 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 a d dimensional real time varying vectors derived from the streams. For a function $f : \mathbf{R}^d \to \mathbf{R}$ we would like to confirm 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

[1]Math. and Stat., UMBC, Baltimore, MD 21250, bmaria2@umbc.edu
[2]Department of Computer Science, Haifa University, Haifa 31905, Israel, dkeren@cs.haifa.ac.il
[3]Math. and Stat., UMBC, Baltimore, MD 21250, kogan@umbc.edu
[4]Math. and Stat., UMBC, Baltimore, MD 21250, yaakovm@umbc.edu

$f > 0$, 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 $f > r$.

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 demostrates that it is impossible to determine from the values of f at the nodes whether its value at the average is above the treshold or not. The present paper deals with the information gain (see Section 2 for details), and rather than focus on the values of f we consider 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

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

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 guarantee the inequality

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

while keeping the nodes silent as much as possible. A possible 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 [5] for a variety of vector norms. The numerical experiments conducted in [5] 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 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.

Clustering in general is a difficult problem, and many clustering problems are known to be NP–complete [1]. Unlike standard clustering that attempts to collect together similar data items [9], we are seeking clusters with dissimilar data items cancelling each other as much as possible. While sub–clusters of a "classical" good cluster usually good, this may not be the case when a cluster contains dissimilar objects.

A basic attempt to cluster nodes was suggested in [6] with results reported for the dataset presented in Section 5. Clustering together just two nodes (the "longest" and "shortest" ones) reported in [6] reduces communication by about 10%.

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

1. We suggest a specific clustering strategy, and report 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. Numerical experiments are reported in Section 5. Section 6 concludes the paper and indicates new research directions. Appendix details accounting of message transmitting.

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 finite text collection (for example a collection of mail or news items). We denote the size of the set $\mathbf{T}$ by $|\mathbf{T}|$. We will be concerned with two subsets of $\mathbf{T}$:

1. $\mathbf{R}$–the set of "relevant" texts (text not labeled as spam),

2. $\mathbf{F}$–the set of texts that contain a "feature" (word or term for example).

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 $\overline{\mathbf{F}} \cap \mathbf{R}$ as follows:

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

Note that

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

The function f is defined on the simplex (i.e. $x_{ij} \geq 0$, $\sum x_{ij} = 1$), 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 easy to see that (4) provides information gain for the "feature" (see e.g. [3]).

As an example we consider n agents installed on n different servers, and a stream of texts arriving at the servers. Let $\mathbf{T}_h = \{\mathbf{t}_{h1}, \ldots, \mathbf{t}_{hw}\}$ 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 $\{x_{ij}(\mathbf{T}_h)\}$, $h = 1, \ldots, n$.

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

The monitoring strategy proposed in [5] can be briefly described as follows:

Algorithm 3.1 *Monitoring Threshold Function*

- *A node is designated as a root* $\mathbf{r}$.
- *The root sets* $i = 0$.

# 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

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

- *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* δ *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* δ *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 local constraint* δ

     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 the experiment) leads to 4006 time instances when 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 constraints violations at exactly one node.

The results immediately suggest to cluster nodes to further reduce communication load. Each cluster will be equipped with a "coordinator" $\mathbf{c}$ (one of cluster's nodes). If a cluster node $\mathbf{n}$ violates its local constraint at time t, then the coordinator collects vectors $\mathbf{v_n}(t) - \mathbf{v_n}(t_i)$ 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. [9]). 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

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

If for each cluster π_i one has $\frac{1}{|\pi_i|}\left\|\sum_{\mathbf{n}\in\pi_i}[\mathbf{v}_{\mathbf{n}}(t)-\mathbf{v}_{\mathbf{n}}(t_j)]\right\|<\delta$, then due to convexity of any norm one has

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

Hence the "new" mean $\frac{1}{n}\sum_{\mathbf{n}\in\mathbf{N}}\mathbf{v}_{\mathbf{n}}(t)$ belongs to $\mathbf{Z}_+(f)$ if the "old" mean $\frac{1}{n}\sum_{\mathbf{n}\in\mathbf{N}}\mathbf{v}_{\mathbf{n}}(t_j)$ belongs to this set. We, therefore, may attempt to define the quality of a k cluster partition Π as

$$Q(\Pi)=\max_i\left\{\frac{1}{|\pi_i|}\left\|\sum_{\mathbf{n}\in\pi_i}[\mathbf{v}_{\mathbf{n}}(t)-\mathbf{v}_{\mathbf{n}}(t_j)]\right\|\right\},\ i=1,\dots,k. \tag{6}$$

Our aim is to identify k and a k cluster partition Π^o that minimizes (6). The monitoring data streams problem requires to assign nodes $\{\mathbf{n}_{i_1},\dots,\mathbf{n}_{i_k}\}$ 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 $\mathbf{v}_{\mathbf{n}}(t)-\mathbf{v}_{\mathbf{n}}(t_j)$ 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. While splitting a "classical" cluster usually generates two clusters of reasonable quality, this may not be the case when a cluster contains dissimilar objects.

The proposed partition quality $Q(\Pi)$ (see (6)) generates three immediate problems:

1. Since the arithmetic mean $\overline{a}$ of a finite set of real numbers $\{a_1,\dots,a_k\}$ satisfies

$$\min\{a_1,\dots,a_k\}\le\overline{a}\le\max\{a_1,\dots,a_k\}$$

   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.

2. Computation of $Q(\Pi)$ involves future values $\mathbf{v}_{\mathbf{n}}(t)$, those are not available at time t_j when clustering is performed.

3. Since the communication overhead of the balancing process is proportional to the size of a cluster, the cluster size should be involved in definition of cluster quality $q(\pi)$.

In the next section we suggest ways to 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 cluster π the size of the cluster also may effect frequency of updates, and, as a result, the communication load. We, therefore, define quality of the cluster π by

$$q(\pi) = \frac{1}{|\pi|} \left|\left| \sum_{\mathbf{n}\in\pi} [\mathbf{v_n}(t) - \mathbf{v_n}(t_j)] \right|\right| + \alpha|\pi|, \tag{7}$$

where α is a nonnegative scalar parameter. The quality of partition $\Pi = \{\pi_1, \ldots, \pi_k\}$ is defined by

$$Q(\Pi) = \max_i q(\pi_i), \tag{8}$$

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

$$0 < \alpha < \max_{\mathbf{n}} ||\mathbf{v_n}(t) - \mathbf{v_n}(t_j)|| . \tag{9}$$

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 $\mathbf{v_n}(t)$ at a future time $t > t_j$ which is not available. While the future is not known, we shall use past values of $\mathbf{v_n}(t)$ for prediction. For each node $\mathbf{n}$ we build "history" vectors $\mathbf{h_n}(t_j)$ defined as follows:

1. $\mathbf{h_n}(t_0) = 0$

2. if ($\mathbf{h_n}(t_j)$ is already available)
   $\mathbf{h_n}(t_{j+1}) = \mathbf{h_n}(t_j)$
   for t increasing from t_j to t_{j+1} do
   $$\mathbf{h_n}(t_{j+1}) = \frac{1}{2}\mathbf{h_n}(t_{j+1}) + [\mathbf{v_n}(t) - \mathbf{v_n}(t_j)]$$

The vectors $\mathbf{h_n}(t_j)$ accumulate history of changes with older changes assigned smaller weights. We shall use the vectors $\{\mathbf{h_n}(t_j)\}$ to generate 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 $h = \max_{\mathbf{n}} ||\mathbf{h_n}(t_j)||$, 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 longest partition cluster π_j, i.e.,

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

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 (7).

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 on account distribution of signals at each node may lead to additional communication savings. We illustrate this statement by a simple example involving just two nodes. If, for example, there is a reason to believe that the inequality

$$2\|\mathbf{v}_1(t)-\mathbf{v}_1(t_i)\| \leq \|\mathbf{v}_2(t)-\mathbf{v}_2(t_i)\| \tag{10}$$

always holds true, 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 signal 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. Assignments of "weighted" local constraints requires information provided by (10). With no additional assumptions about signal distribution this information is not available. Unlike [4] we refrain from making assumptions regarding possible 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

$$w_1=\ldots=w_n=1\ \text{ and } W_1=\ldots=W_n=1\ (\text{so that } \sum_{j=1}^{n}w_j=\sum_{j=1}^{n}W_j=n). \tag{11}$$

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 $w_j = n \times \dfrac{W_j}{W}$ (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 $\delta(\mathbf{c}) = \dfrac{1}{|\pi|} \sum_{\mathbf{n} \in \pi} \delta(\mathbf{n})$.

5 Numerical Experiments

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 6700 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). For the experiments described below the threshold value r is

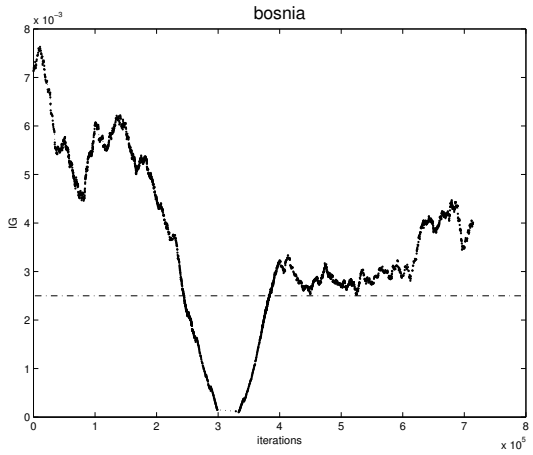

Figure 1: information gain values for the feature "bosnia"

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 [5] with the feature "febru," and the threshold $r = 0.0025$ are shown in Table 2 where

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

Table 2: number of mean computations, and broadcasts for feature "febru" with threshold $r = 0.0025$, no 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

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

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 $\alpha = 0.05, 0.10, \ldots, 0.95$. The best result with respect to α is 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 drops down by about 50%.

Next we turn to the features "ipo" and "bosnia." In both cases we run monitoring with clustering letting $\alpha = 0.05, 0.10, \ldots, 0.95$ 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.

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 [5] are presented in Table 6. Application of the clustering procedure leads to a slight decrease in

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

Table 4: number of mean computations, and broadcasts for feature "ipo" with threshold $r = 0.0025$, no 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 5: number of root and coordinator mean computations, and total broadcasts for feature "ipo" with threshold $r = 0.0025$ with clustering

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

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

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 is increasing. Clustering is no universal remedy, in some cases better performance is achieved with no clustering (by keeping α between 0.05 and 0.95 we force nodes to get together into clusters).

6 Conclusions and future research directions

In this paper we propose to monitor threshold function over distributed data streams through clustering nodes that cancel each other. The strategy, if successful, in many cases limits communications required to message exchanges within a cluster only, and brings communication savings.

The particular 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" constant $0 < \alpha < 1$ selected at the beginning of the monitoring

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

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

process. While the results obtained show improvement over previously reported ones that do not use clustering [5] it is of interest to introduce an update of α based on the monitoring history each time nodes are re–clustered (see e.g. [12] 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.

References

[1] P. Brucker. On the complexity of clustering problems. In *Lecture Notes in Economics and Mathematical Systems, Volume 157*, pages 45–54. Springer-Verlag, Berlin, 1978.

[2] M Dilman and D. Raz. Efficient reactive monitoring. In *INFOCOM '01*, pages 1012–1019. Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communication Societies, 2001.

[3] R.M. Gray. *Entropy and Information Theory*. Springer–Verlag, New York, 1990.

[4] D. Keren, I. Sharfman, A. Schuster, and A. Livne. Shape sensitive geometric monitoring. *IEEE Transactions on Knowledge and Data Engineering*, 24(8):1520–1535, 2012.

[5] J. Kogan. Feature selection over distributed data streams through convex optimization. In *Proceedings of the Twelfth SIAM International Conference on Data Mining (SDM 2012)*, pages 475–484, Anaheim, CA, USA, 2012. SIAM.

[6] J. Kogan and Y. Malinovsky. 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)*, pages 5–13, Austin, TX, USA, 2013. SIAM.

[7] S. Madden and M.J. Franklin. An architecture for queries over streaming sensor data. In *ICDE 02*, page 555, Washington, DC, USA, 2002. IEEE Computer Society.

[8] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. Finding (recently) frequent items in distributed data streams. In *ICDE 05*, pages 767–778, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[9] B. Mirkin. *Clustering for Data Mining: A Data Recovery Approach*. Chapman & Hall/CRC, Boca Raton, 2005.

[10] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Transactions on Database Systems*, 32(4):23:1–23:29, 2007.

[11] I. Sharfman, A. Schuster, and D. Keren. A Geometric Approach to Monitoring Threshold Functions over Distributed Data Streams. In M. May and L. Saitta, editors, *Ubiquitous Knowledge Discovery*, pages 163–186. Springer–Verlag, 2010.

[12] J.C. Willems. *The Analysis of Feedback Systems*. The MIT Press, Cambridge, Mass., 1971.

[13] B.-K. Yi, Sidiropoulos N., Johnson T., H.V. Jagadish, C. Faloutsos, and Biliris A. Online datamining for co–evolving time sequences. In *ICDE 00*, page 13, Washington, DC, USA, 2000. IEEE Computer Society.

[14] Y. Zhu and D. Shasha. Statestream: Statistical monitoring of thousands of data streamsin real time. In *VLDB*, pages 358–369, 2002.

Appendix–Counting Broadcasts

Transmission of a double precision real number is defined as a message in [5]. In this paper, in addition to real numbers typically representing vector coordinates, integer values such as node ID and node "reporting order" should also be transmitted. Transmission of node IDs is needed, for example, to allow the root to cluster nodes. To minimize communication load nodes in smaller clusters report violations of node constraints first, and the reporting order is assigned and communicated to nodes by the root that knows all cluster sizes.

Since every vector $\mathbf{v}$ associated with a node belongs to a simplex it is represented by a real number not exceeding 1. We may use the integer part of these real numbers for transmission of integers. There is a variety of coding and compression techniques that can be used to transmit a set of real numbers as a single real. The discussion of these methods is beyond the scope of this paper. In order to be able to compare different monitoring techniques we shall count a number of broadcasts, where by a broadcast we mean a single communication between two nodes. As an illustration below we compute the number of broadcasts needed for one iteration of Algorithm 3.1 triggered by violation of a node constraint. We first assume that the violator node $\mathbf{n}$ is different from the root.

1. The violator node $\mathbf{n}$ notifies all other nodes (except the root) about the violation ($n-2$ broadcasts).

2. Each node $\mathbf{n}$ broadcasts its vector $\mathbf{v_n}$ to the root ($n-1$ broadcasts).

3. The root recomputes $\delta(\mathbf{r})$ and sends it to each node ($n-1$ broadcasts).

This leads to $3(n-1)-1$ broadcasts. If the violator node $\mathbf{n}$ is the root itself the number of broadcasts becomes $3(n-1)$ (at step 1 above the root has to make $n-1$ broadcasts).

Next we turn to the monitoring with clustering. The monitoring procedure starts with each node $\mathbf{n}$ sending it's initial vector $\mathbf{v_n}(t_0)$ to the root $\mathbf{r}$ (that requires $n-1$ broadcasts). The root computes the mean $\frac{1}{n}\sum_{\mathbf{n}} \mathbf{v_n}(t_0)$ of the initial vectors, computes $\delta(\mathbf{r})$, and broadcasts $\delta(\mathbf{r})$ to each node ($n-1$ broadcasts). After exchanging

$$2(n-1) \tag{12}$$

broadcasts the monitoring proceeds with each node being a singleton cluster.

1. As long as the inequality

   $$|\mathbf{v_n}(t) - \mathbf{v_n}(t_0)| < \delta(\mathbf{r}) \text{ holds true for each node } \mathbf{n}$$

   the nodes are silent. At the first time instance t when the inequality fails for at least one node $\mathbf{n}$ the following actions are triggered:

   (a) the node $\mathbf{n}$ (if the node itself is not the root) broadcasts it's ID and vector $\mathbf{v_n}(t)$ to the root (1 broadcast),

   (b) the root issues $n-2$ requests for ID and $\mathbf{v_n}(t)$ to the other nodes ($n-2$ broadcasts),

   (c) $n-2$ nodes report their IDs and $\mathbf{v_n}(t)$ vectors to the root ($n-2$) broadcasts),

   This bring the number of broadcasts to $2n-3$. If the node violator is the root, then this number is $2n-2$. To simplify the computations we select the largest number $2n-2$.

   At this step keeping in mind (12) the total number of broadcasts needed to be exchanged is

   $$2(n-1) + 2n - 2 = 4(n-1). \tag{13}$$

2. Next the root recomputes $\delta(\mathbf{r})$, clusters nodes, and broadcasts to each node ($n-1$ broadcasts) its updated local constraint $\delta(\mathbf{n})$, the ID of it's coordinator, and the reporting order. If a node is also a coordinator, then IDs of its nodes, and coordinator reporting order are provided to the coordinator by the root. Keeping in mind (13) the total number of broadcasts right after the first root mean update and first clustering is

   $$5(n-1). \tag{14}$$

Clusters are now formed, and we shall count the number of broadcasts needed to be exchanged for each of the three types of possible violations:

1. A node constraint is violated in a singleton cluster.

   (a) the violator node $\mathbf{n}$ reports it's ID, $\mathbf{v_n}(t)$, $W_\mathbf{n}$, and the history vector $\mathbf{h_n}$ to the root (1 broadcasts),

   (b) the root requests all other $n-2$ nodes to provide their input (ID's, $\mathbf{v_n}(t)$ vectors, $W_\mathbf{n}$ weights, and history vectors $\mathbf{h}$, total of $n-2$ broadcasts),

   (c) the $n-2$ nodes report ID's, $\mathbf{v_n}(t)$ vectors, $W_\mathbf{n}$ weights, and history vectors $\mathbf{h}$ to the root ($(n-2)$ broadcasts),

   (d) the root recomputes the constraint $\delta(\mathbf{r})$, node constraints $\delta(\mathbf{n})$, and reports to each node it's coordinator ID, $\delta(\mathbf{n})$, and the node "reporting order." Cluster coordinators also receive IDs of the nodes in their respective clusters ($n-1$ broadcasts).

   This leads to $3(n-1)-1$ broadcasts if violator node is no root, and $3(n-1)$ broadcasts if violation is committed by the root. To compute the broadcasts we use the larger number

$$3(n-1). \tag{15}$$

2. A node constraint is violated in a non singleton cluster π with coordinator $\mathbf{c}$.

   (a) the violator $\mathbf{n}$ reports it's ID, $\Delta_\mathbf{n}$, and $\delta_\mathbf{n}$ to the coordinator $\mathbf{c}$ (1 broadcast),

   (b) the coordinator $\mathbf{c}$ sends request for $\Delta_\mathbf{n}$ vectors and node constraints $\delta_\mathbf{n}$ for all nodes in its cluster π other then $\mathbf{n}$ and itself ($|\pi|-2$ broadcasts)

   (c) the nodes broadcast their vectors Δ and constraints δ to the coordinator (total of $(|\pi|-2)$ broadcasts). The total comes to $2|\pi|-3$, and this number is $2|\pi|-2$ when violator node is the coordinator.

   The total of broadcasts needed is:

$$2|\pi|-2. \tag{16}$$

3. A coordinator constraint is violated. First we assume the coordinator $\mathbf{c}$ is different from the root:

   (a) the coordinator $\mathbf{c}$ of cluster π broadcasts requests to all nodes (except itself and the root) to provide the root with their IDs, vectors $\mathbf{v_n}(t)$, weights W, and history vectors $\mathbf{h}$ ($n-2$ broadcasts).

   (b) $n-1$ nodes ($n-2$ nodes requested by the coordinator and the coordinator itself) send the requested information to the root ($n-1$ broadcasts).

(c) the root recomputes $\delta(\mathbf{r})$, clusters nodes and provides each node with updated local constraint $\delta(\mathbf{n})$, the new cluster affiliation (i.e. ID of a new coordinator), and the node "reporting order." Coordinators are also provided with the IDs of their nodes (total of $n-1$ broadcasts).

This brings the number of broadcasts to $3(n-1)-1$. If $\mathbf{c}$ is the root, then this number is $3(n-1)$, and this is the number we use to compute broadcasts

$$3(n-1). \tag{17}$$