Adaptive Clustering for Monitoring Distributed Data Streams

Maria Barouti* Daniel Keren† Jacob Kogan‡ Yaakov Malinovsky§

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. The proposed approach enables monitoring values of a threshold function over distributed data streams through a set of constraints applied independently on each cluster of streams. The clusters are designed to adapt themselves to the data stream. We report experiments with clustering approach that yield communication load reduction.

Keyword list: adaptive stream mining, convex analysis, distributed system, clustering

1 Introduction

In many emerging applications one needs to process a continuous stream of data in real time. The current contribution is motivated by results reported in [4], 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

$$f\left(\frac{\mathbf{v}_1(t) + \ldots + \mathbf{v}_n(t)}{n}\right) > 0 \tag{1.1}$$

while minimizing communication between the nodes. In e.g. [4, 3] a few real-life applications of this monitoring problem are described; see also Section 2 here.

The present paper deals with the information gain function (see Section 2 for details). Rather than focus on the values of f we consider the location of the vectors $\mathbf{v}_i(t)$ relative to the set $\mathbf{Z}_+(f) = \{\mathbf{v} : f(\mathbf{v}) > 0\}$. We restate (1.1) as

$$\mathbf{v}(t) = \frac{\mathbf{v}_1(t) + \ldots + \mathbf{v}_n(t)}{n} \in \mathbf{Z}_+(f). \tag{1.2}$$

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 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 $v(t_0) = \frac{v_1(t_0) + v_2(t_0) + v_3(t_0)}{3} > 0$ 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 [5]. The numerical experiments conducted in [5] with the dataset described in Section 5 show that (a) the number of time instances the mean violates (1.1) is a small fraction ($< 1\%$) of the number of time instances when the local constraint is violated at the nodes, (b) the lion's share of communications (about 75%) is required because of a single node violation of the local constraint δ.

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

$$v_1(t) - v_1(t_0) = -[v_2(t) - v_2(t_0)], \text{while } |v_3(t) - v_3(t_0)| < \delta$$

then $|v(t) - v(t_0)| < \delta$, $f(v(t)) > 0$, and update of the mean can be avoided. Separate monitoring of the two

*Math. and Stat., UMBC, Baltimore, MD 21250, bmaria2@umbc.edu

†Department of Computer Science, Haifa University, Haifa 31905, Israel, dkeren@cs.haifa.ac.il

‡Math. and Stat., UMBC, Baltimore, MD 21250, kogan@umbc.edu

§Math. and Stat., UMBC, Baltimore, MD 21250, yaakovm@umbc.edu

node cluster $\{\mathbf{n}_1, \mathbf{n}_2\}$ would require communication involving two nodes only, and could reduce communication load. In this paper we advance clustering approach to monitoring. A specific clustering strategy applicable with a variety of norms leading to communication savings is the main contribution of this work.

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.

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 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" (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:

$$x_{11}(\mathbf{T}) = \frac{|\mathbf{F} \cap \overline{\mathbf{R}}|}{|\mathbf{T}|}, \quad x_{12}(\mathbf{T}) = \frac{|\mathbf{F} \cap \mathbf{R}|}{|\mathbf{T}|}, \tag{2.3}$$
$$x_{21}(\mathbf{T}) = \frac{|\overline{\mathbf{F}} \cap \overline{\mathbf{R}}|}{|\mathbf{T}|}, \quad x_{22}(\mathbf{T}) = \frac{|\overline{\mathbf{F}} \cap \mathbf{R}|}{|\mathbf{T}|}.$$

Note that $0 \le x_{ij} \le 1$, and $x_{11} + x_{12} + x_{21} + x_{22} = 1$. The function f is given by

$$\sum_{i,j} x_{ij} \log \left(\frac{x_{ij}}{(x_{i1} + x_{i2})(x_{1j} + x_{2j})} \right), \tag{2.4}$$

where $\log x = \log_2 x$ throughout the paper. The *information gain* for the "feature" is provided by f [1].

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

3034	620	162	70	38	26	34	17	5	0
1	2	3	4	5	6	7	8	9	10

Table 1: first row–number of time instances when a local constraint has been violated by exactly k nodes, $k = 1, 2, \ldots, 10$; second row–number of nodes–violators, $r = 0.0025$, l_2 norm, the feature is "bosnia"

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

$$f(x_{11}(\mathbf{T}), x_{12}(\mathbf{T}), x_{21}(\mathbf{T}), x_{22}(\mathbf{T})) - r > 0 \tag{2.5}$$

while minimizing communication between the servers.

3 Clustering for monitoring: motivation

In what follows we denote a norm of a vector $\mathbf{v}$ by $\|\mathbf{v}\|$. A specific choice of a norm will be indicated when needed.

The monitoring strategy proposed in [5] and applied to data streams generated from the data described in Section 5 leads to 4006 time instances in which the local constraints are violated, and the root is updated. Table 1 shows that in 3034 out of 4006 time instances, communications with the root are triggered by constraint violations at exactly one node.

The results suggest to cluster nodes to reduce communication load. Each cluster π will be equipped with a "coordinator" $\mathbf{c}$. If $\mathbf{n}' \in \pi$ violates its local constraint at time t, then the coordinator collects vectors $\mathbf{v_n}(t) - \mathbf{v_n}(t_i)$ from all $\mathbf{n} \in \pi$, computes the mean, and checks whether the mean violates the coordinator constraint $\delta(\mathbf{c})$. We shall follow [4] and refer to this step as "the balancing process." If $\delta(\mathbf{c})$ is violated, the mean of the entire dataset is recomputed.

A standard clustering problem is often described as "...finding and describing cohesive or homogeneous chunks in data, the clusters" [2]. For the problem at hand we would like to partition the set of nodes $\mathbf{N}$ into k disjoint clusters $\Pi = \{\pi_1, \ldots, \pi_k\}$. If $\frac{1}{|\pi_i|} \left\| \sum_{\mathbf{n} \in \pi_i} [\mathbf{v_n}(t) - \mathbf{v_n}(t_j)] \right\| < \delta$ for each i, then one has

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

Hence the "new" mean $\frac{1}{n} \sum_{\mathbf{n} \in \mathbf{N}} \mathbf{v_n}(t)$ belongs to $\mathbf{Z}_+(f)$ if the "old" mean $\frac{1}{n} \sum_{\mathbf{n} \in \mathbf{N}} \mathbf{v_n}(t_j)$ belongs to this set. A

possible definition for quality of a partition Π is

$$(3.6)\quad Q(\Pi)=\max_{i\in\{1,\ldots,k\}}\left\{\frac{1}{|\pi_i|}\left|\left|\sum_{\mathbf{n}\in\pi_i}[\mathbf{v_n}(t)-\mathbf{v_n}(t_j)]\right|\right|\right\}$$

Our aim is to identify k and a k cluster partition Π^o that **minimizes** (3.6). 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

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

is minimized. Hence, unlike classical clustering procedures, this one needs to combine "dissimilar" nodes together.

The proposed partition quality $Q(\Pi)$ (see (3.6)) generates three immediate problems:

1. The single cluster partition always minimizes $Q(\Pi)$.
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. The individual clusters' sizes should affect the clustering quality $q(\pi)$ to account for the balancing process communication.

4 Clustering for monitoring: implementation

We define the quality of the cluster π by

$$(4.7)\quad q(\pi)=\frac{1}{|\pi|}\left|\left|\sum_{\mathbf{n}\in\pi}[\mathbf{v_n}(t)-\mathbf{v_n}(t_j)]\right|\right|+\alpha|\pi|,$$

where $\alpha\geq 0$ is a scalar parameter. The quality of the partition Π is defined by

$$(4.8)\qquad Q(\Pi)=\max\{q(\pi_i)\ :\ i=1,\ldots,k\}.$$

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

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. We shall use past values of $\mathbf{v_n}(t)$ for prediction. For each node $\mathbf{n}$ we build "history" vectors $\mathbf{h_n}(t_j)$ that accumulate the weighted history of changes, with the current change carrying a weight twice as much as the previous one. When the vector set is normalized by the magnitude of the longest vector in the set, the range for α conveniently shrinks to $[0,1]$, and the induced optimal partitioning remains the same. In what follows we set $h=\max_{\mathbf{n}\in\mathbf{N}}||\mathbf{h_n}(t_j)||$, assume that $h>0$, and describe a 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.
2. Identify cluster π_i so that the merger of π_i with π_j produces a cluster of smallest possible quality (4.7).

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.

If there is reason to believe that e.g. the following inequality

$$(4.9)\qquad 2\|\mathbf{v}_1(t)-\mathbf{v}_1(t_i)\|\leq\|\mathbf{v}_2(t)-\mathbf{v}_2(t_i)\|$$

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

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

holds true. Assigning "weighted" local constraints requires information provided by (4.9). With no additional assumptions about the stream data distribution this information is not available. We estimate the weights through past values $\|\mathbf{v}_j(t)-\mathbf{v}_j(t_i)\|$.

1. Start with the initial set of weights

$$(4.10)\quad w_1=\ldots=w_n=1,\ W_1=\ldots=W_n=1.$$

2. At the next time t, each node $\mathbf{n}_j$ computes updates

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

Next time when the distance δ from the mean to the boundary of $\mathbf{Z}_+(f)$ is 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 $\delta(\mathbf{n}_j)=w_j\delta$ where $w_j=n\times\frac{W_j}{W}$ back to node j. For a coordinator $\mathbf{c}$ of a node cluster π the constraint $\delta(\mathbf{c})=\frac{1}{|\pi|}\sum_{\mathbf{n}\in\pi}\delta(\mathbf{n})$.

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

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

Table 4: Number of mean computations, and broadcasts, for feature "bosnia" with threshold $r = 0.0025$, no 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

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

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.

Each document is labeled as belonging to one or more categories. We follow [4] and label a vector as "relevant" if it belongs to the "CORPORATE/INDUSTRIAL" ("CCAT") category, and "spam" otherwise. Following [4] 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. 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.

The numerical experiment reported in [5] 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 for the same feature and threshold with $\alpha = 0.05, 0.10, \ldots, 0.95$. The best result for l_1, l_2, and l_∞ norms with respect to α are presented in Table 3, and the result shows about 50% decrease in the number of broadcasts.

Next we turn to the features "ipo" and "bosnia." While communication savings for "ipo" are similar to those presented above for "febru", an application of clustering to monitoring "bosnia's" information gain appears to be far less successful. Results obtained without clustering in [5] are presented in Table 4. 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 5). In case of the l_1 norm, the number of broadcasts increases. This is just a reminder that clustering is no universal remedy, and in some cases better performance is achieved with no clustering.

References

[1] Gray, R.M.: Entropy and Information Theory. Springer–Verlag, New York, (1990)

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

[3] Gabel, M. and Schuster, A. and Keren, D.: Communication-efficient outlier detection for scale-out systems. In BD3@VLDB, 19-24, (2013)

[4] I. Sharfman and A. Schuster and D. Keren: A Geometric Approach to Monitoring Threshold Functions over Distributed Data Streams, ACM Transactions on Database Systems, **32**, 23:1-23:29, (2007)

[5] Kogan, J.: Feature Selection over Distributed Data Streams through Convex Optimization. Proceedings of the Twelfth SIAM International Conference on Data Mining (SDM 2012), SIAM, 475-484, (2012)