APPROVAL SHEET

Title of Dissertation: Clustering for Monitoring Distributed Data Streams

Name of Candidate: Maria Barouti
Doctor of Philosophy, 2016

Dissertation and Abstract Approved:

Jacob Kogan
Professor of Mathematics
Department of Mathematics and Statistics

Yaakov Malinovsky
Assistant Professor of Statistics
Department of Mathematics and Statistics

Date Approved: _____

ABSTRACT

Title of dissertation: CLUSTERING FOR MONITORING DISTRIBUTED DATA STREAMS

Maria Barouti
Doctor of Philosophy, 2016

Dissertation directed by: Jacob Kogan
Professor of Mathematics
Department of Mathematics and Statistics
University of Maryland, Baltimore County

Yaakov Malinovsky
Assistant Professor of Statistics
Department of Mathematics and Statistics
University of Maryland, Baltimore County

Data mining is a challenging research area of computer science with profound applications in database industries and resulting market needs. Data mining is the computational process of discovering patterns in big data sets. This process enables us to extract valuable information from large data by involving methods at the intersection of different topics such as machine learning, statistics, and artificial intelligence. Over the last years there has been a growing interest in data analysis research by monitoring data streams in a distributed system. In this study we propose to monitor arbitrary threshold functions over distributed data streams while minimizing communication overhead. To illustrate this further, assume that we have a number of sensors that are spread in the space and we would like to monitor the average of their measurements while minimizing communication between the sensors. Each sensor represents a node that produces time varying vectors derived from the stream of measurements. Thus we are interested to check if a function evaluated at the vectors' average at each time is greater than zero while communication between the nodes is minimized.

Motivated by recent contributions based on geometric ideas, and after reviewing some well known clustering algorithms we present an alternative approach that combines system theory techniques, clustering and statistical approaches. Our 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 evolve in time and 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, subclusters 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/or modification of the existing ones, and this thesis is a step in this direction (extension of a book chapter, see Barouti et al. [2015]).

We report experiments on real-world data with a newly devised clustering algorithm. The experiments detect instances where communication between nodes is required, and show that the clustering approach reduces communication load. Last but not least, we indicate new future directions and discuss possible methodologies that can be involved into my future research agenda.

CLUSTERING FOR MONITORING DISTRIBUTED DATA STREAMS

by

Maria Barouti

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, Baltimore County in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2016

Advisory Committee:
Dr. Jacob Kogan, Professor/Advisor
Dr. Yaakov Malinovsky, Assistant Professor/Co-Advisor
Dr. James Lo, Professor
Dr. Charles Nicholas, Professor
Dr. Junyong Park, Associate Professor

© Copyright by
Maria Barouti
2016

DEDICATION

To my parents, Athanasios and Marina and my siblings, Nikolaos and Katerina. Last but not least, to my partner in life and science Zois Boukouvalas.

ACKNOWLEDGMENTS

It has been a great privilege to study mathematics at UMBC. Having previously completed a Bachelors degree in Mathematics and a Masters degree in Applied and Computational Mathematics, I deeply appreciated the chance I was given by UMBC to broaden my knowledge in Mathematics by obtaining a Masters degree as well as pursuing a PhD in Applied Mathematics.

Firstly, I would like to express my sincere gratitude to my advisors and mentors, Dr. Jacob Kogan and Dr. Yaakov Malinovsky, who have been a great inspiration and source of enthusiasm for me. Dr. Kogan and Dr. Malinovsky were the people who mostly influenced my academic career and guided me through my journey in the broad area of data mining and machine learning. Their steady encouragement and help was a cornerstone for completing this thesis. Furthermore, their advice on both research as well as my career have been priceless allowing me to grow as a research scientist.

Furthermore, I would like to thank Drs. Charles Nicholas, James Lo and Junyong Park for graciously serving in my dissertation committee and for providing valuable feedback about the work. The UMBC Department of Mathematics and Statistics has been a wonderful, supportive environment for graduate study. Thanks to Janet Burgee, Deneen Blair, Boris Alemi, Marshal Turner, Maggie Kennedy and all the staff who helped me during my time in the department. Thanks to all the faculty for their teaching and guidance, especially the mathematics group: Drs. Rouben Rostamian, Muddappa Gowda, Muruhan Rathinam, Kathleen Hoffman, Raji Baradwaj, Kalman Nanes, and Liz Stanwyck. I would also like to thank my previous supervisor Dr. Andreas Arvanitoyergos, for his encouragement over the years.

Special thanks to my family and friends for their endless support. Finally and most importantly for his many years love and ability to keep me grounded, I would like to thank Zois Boukouvalas.

Contents

Chapter 1

Introduction

In many emerging applications such as sensor networks (Madden and Franklin [2002]), network monitoring (Dilman and Raz [2002]) etc. one needs to process a continuous stream of data in real time. Monitoring queries are a particular class of queries in the context of data streams. Previous work in this area examine monitoring simple aggregates (Dilman and Raz [2002]), or term frequency occurrence in a set of distributed streams (Manjhi et al. [2005]) where the algorithms proposed enable detecting when the sum of a distributed set of variables exceeds a predetermined threshold. However, these algorithms concentrate on monitoring the sum of a set of variables, whereas in this thesis enables monitoring values of an arbitrary threshold function. A useful, 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 $\mathbf{N} = \{\mathbf{n}_1, \ldots, \mathbf{n}_n\}$ and 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

$$f\left(\frac{\mathbf{v}_1(t) + \cdots + \mathbf{v}_n(t)}{n}\right) > 0 \tag{1.0.1}$$

while minimizing communication between the nodes. We refer to this query as a threshold function query. 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 $f > r$.

A task of feature selection is an important task that requires very high communication overhead using naive, centralized algorithms. Motivated by results reported in (Sharfman et al. [2007]) we are interested in determining, at any given time, whether the value of an arbitrary function on the average of these vectors crosses a predetermined threshold. In the next section we present a Text Mining application that consists a relevant example of the monitoring query described above.

1.1 Text Mining Application

Information and data are generated everyday through economic, academic and social activities. Techniques such as text and data mining are required to exploit a significant potential economic and societal value. Text mining is an interdisciplinary field that draws on information retrieval, data mining, machine learning, statistics, and computational linguistics. Text mining is the process of unlocking hidden information in plain sight as well as developing new knowledge by finding unseen connections and patterns through countless pages of digitized text. Since the vocabulary comprising data streams may be very large, an important task is to determine which words, or features, should be used for performing a classification. This task is known as feature selection and it is performed as follows:

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:

$$\begin{aligned} x_{11}(\mathbf{T}) &= \tfrac{|\mathbf{F} \cap \overline{\mathbf{R}}|}{|\mathbf{T}|},\ x_{12}(\mathbf{T}) = \tfrac{|\mathbf{F} \cap \mathbf{R}|}{|\mathbf{T}|}, \\ x_{21}(\mathbf{T}) &= \tfrac{|\overline{\mathbf{F}} \cap \overline{\mathbf{R}}|}{|\mathbf{T}|},\ x_{22}(\mathbf{T}) = \tfrac{|\overline{\mathbf{F}} \cap \mathbf{R}|}{|\mathbf{T}|}. \end{aligned} \tag{1.1.1}$$

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

$$\sum_{i,j} x_{ij} \log \left(\frac{x_{ij}}{(x_{i1} + x_{i2})(x_{1j} + x_{2j})} \right), \tag{1.1.2}$$

where $\log x = \log_2 x$ throughout the thesis. The *information gain* for the "feature" is provided by f (see e.g. (Gray [2011]).

As an example of monitoring a continuous query, 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

$$f(\mathbf{v}) > 0, \tag{1.1.3}$$

where $\mathbf{v} = (x_{11}(\mathbf{T}), x_{12}(\mathbf{T}), x_{21}(\mathbf{T}), x_{22}(\mathbf{T}))$ while minimizing communication between the servers.

Thus all the features scoring above a certain threshold can be chosen as parameters for the classification task. Moreover, the example presented below shows that there is a fundamental difference between monitoring the cases of linear and non-linear f.

Let $f(x)$ be a linear function and $\mathbf{v}_i$, $i = 1, 2$ are scalar values stored at two distinct nodes. Then if both values of f at nodes have the same sign its value at the average will have this sign too, since $f(\frac{\mathbf{v}_1+\mathbf{v}_2}{2}) = \frac{1}{2}f(\mathbf{v}_1) + \frac{1}{2}f(\mathbf{v}_2)$. However, this is not true when f is not linear. To see that let $f(x) = x^2 - 6$. Note that if $\mathbf{v}_1 = -4$, and $\mathbf{v}_2 = 4$, then

$$f(\mathbf{v}_1) = f(\mathbf{v}_2) = 10 > 0 \text{ and } f(\tfrac{\mathbf{v}_1+\mathbf{v}_2}{2}) = -6 < 0.$$

If $\mathbf{v}_1 = -2$, and $\mathbf{v}_2 = 4$, then

$$f(\mathbf{v}_1) = -2 < 0,\ f(\mathbf{v}_2) = 10 > 0 \text{ and } f(\tfrac{\mathbf{v}_1+\mathbf{v}_2}{2}) = -5 < 0.$$

Finally, when $\mathbf{v}_1 = 2$, and $\mathbf{v}_2 = 4$ one has

$$f(\mathbf{v}_1) = -2 < 0,\ f(\mathbf{v}_2) = 10 > 0 \text{ and } f(\tfrac{\mathbf{v}_1+\mathbf{v}_2}{2}) = 3 > 0.$$

This thesis is handling non linear threshold functions, with a specific focus on information gain (1.1.2).

1.2 Related Work

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 (Sharfman et al. [2007]). The difficulty of determining from the values of f at the nodes whether its value at the average is above a threshold or not led the authors of (Sharfman et al. [2007]) to present a distributed algorithm for locally determining whether f's value at the average data vector is above the threshold or not. The geometric approach presented in (Sharfman et al. [2007]) allows splitting an arbitrary monitoring task into a set of constraints that applied locally on each of the streams.

Specifically, the proposed solution is to monitor the values of f on the convex hull of $\{\mathbf{v}(t_i), \mathbf{u}_1(t), \mathbf{u}_2(t), ...\mathbf{u}_n(t)\}$ where $\mathbf{u}_j(t) = \mathbf{v}(t_i) + [\mathbf{v}_j(t) - \mathbf{v}_j(t_i)]$, $t \geq t_i$ and $\mathbf{v}(t_i) = \frac{1}{n}\sum_{j=1}^{n} \mathbf{v}_j(t_i)$. This technique is based on two observations:

- The convexity property: the mean $\mathbf{v}(t)$ is in the convex hull of $\{\mathbf{v}(t_i), \mathbf{u}_1(t), \mathbf{u}_2(t), ...\mathbf{u}_n(t)\}$. (Note that $\mathbf{u}_j(t)$ and $\mathbf{v}(t_i)$ are available to node j without any communication with other nodes).

- The union of the l_2 balls $B_2(\mathbf{v}, \mathbf{u}_j)$ of radius $\frac{1}{2}||\mathbf{v} - \mathbf{u}_j||$ centered at $\frac{\mathbf{v}+\mathbf{u}_j}{2}$ covers the convex hull of the vectors (ball cover), i.e. conv$\{\mathbf{v}(t_i), \mathbf{u}_1(t), \mathbf{u}_2(t), ...\mathbf{u}_n(t)\} \subseteq \cup B_2(\mathbf{v}(t_i), \mathbf{u}_j(t))$ (see Figure 1.2.1). The triangle denotes the convex hull of the vectors.

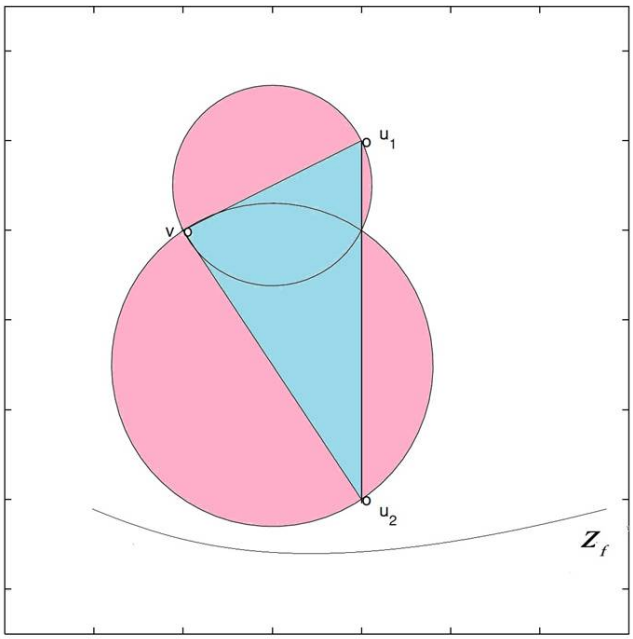


Fig. 1.2.1: Ball cover where $\mathbf{Z}_f = \{x \ : \ f(x) = 0\}$

Since each ball $B_2(\mathbf{v}(t_i), \mathbf{u}_j(t))$, $t \geq t_i$, $j = 1, 2, ..., n$ can be monitored by node j with no communication with other nodes, the second observation allows to split monitoring of the convex hull into n independent tasks executed by the n nodes separately (ball monitoring) and without communication. Therefore, instead of collecting all vectors to a central location which is very costly in terms of communication load, (Sharfman et al. [2007]) chooses to impose numerical constraints on the data collected at each node. The local constraint on each stream is set as follows: the monitored function f and the thresh-

old can be seen as inducing a coloring over $\mathbf{R}^d$. Thus the vectors $\{x \ : \ f(x) > 0\}$ are said to be different color from the vectors $\{x \ : \ f(x) \leq 0\}$. The local constraints that all nodes maintain is to check if all the l_2 balls described above are of the same color (pink) as shown in Figure 1.2.1. Note that the convex hull of the vectors is denoted by the blue triangle.

The geometric analysis of this problem guarantees that as long as the constraints on all the streams are upheld, the result of the query remains unchanged and thus no communication is required. If a constraint in one of the streams is violated (see Figure 1.2.2) then new data is gathered from the streams, the query is reevaluated, and new constraints are set on the streams. This strategy leads to sufficient conditions for (1.0.1), and may be conservative.

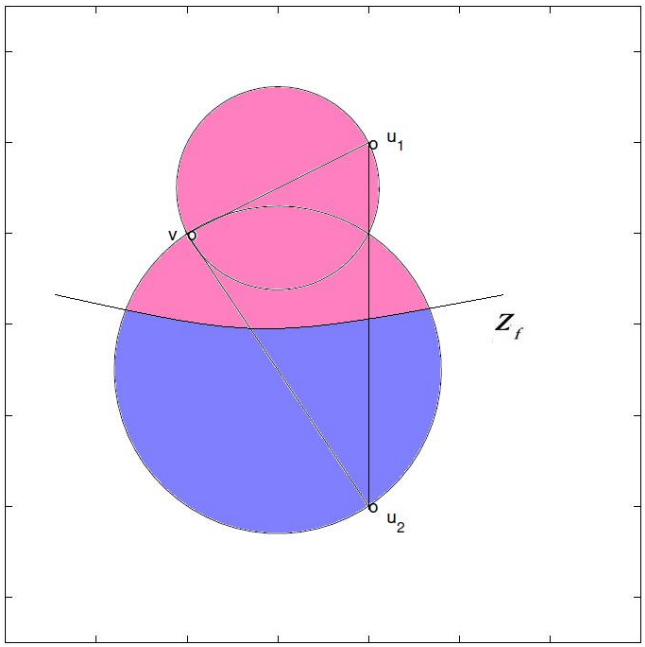


Fig. 1.2.2: Violation at node $\mathbf{n}_2$

A more recent work reported in (Kogan [2012]) is motivated by results reported in (Sharfman et al. [2007]). In (Kogan [2012]) the setting of the problem is the same (deals with the information gain function) but here the author in order to check if the mean $\mathbf{v}(t)$ belongs to the set $\mathbf{Z}_+(f) = \{x \ : \ f(x) > 0\}$, he focuses on the distance from the mean to the boundary of the set $\mathbf{Z}_+(f)$. Note that its boundary denoted by $\mathbf{Z}_f$. Thus the functional

monitoring problem (1.0.1) is transformed to the monitoring of the following geometric condition

$$\mathbf{v}(t) = \frac{\mathbf{v}_1(t) + \cdots + \mathbf{v}_n(t)}{n} \in \mathbf{Z}_+(f). \tag{1.2.1}$$

The author extends the idea of monitoring and provides a threshold monitoring algorithm which is described as follows.

At time t_0 the vectors $\mathbf{v}_i(t_0)$ for $i = 0, 1, ..., n$, the mean $\mathbf{v}(t_0)$ and the local constraint $\delta = \text{dist}\,(\mathbf{v}(t_0), \mathbf{Z}_f)$ are computed. The local constraint δ is made available to all nodes. As new vector $\mathbf{v}_k(t)$ is computed at some node $\mathbf{n}_k$ at time t, the inequality $||\mathbf{v}_k(t_0) - \mathbf{v}_k(t)|| < \delta$ is checked. If the inequality holds true then the node keeps silent, no updates of the mean $\mathbf{v}(t_0)$ and the local constraint δ are required. If the inequality fails at time t at least at one node then the nodes communicate, the mean $\mathbf{v}(t)$ is updated, the new local constraint $\delta = \text{dist}(\mathbf{v}(t), \mathbf{Z}_f)$ is computed and made available to each node. This process continues until the end of the stream. Thus in (Kogan [2012]) only a scalar δ should be communicated to each node as opposed to (Sharfman et al. [2007]) where the value of the updated mean should be transmitted and this can be costly for high dimensional problems.

This type of monitoring suggested in (Kogan [2012]) allows to apply the above monitoring algorithm to a variety of distance functions (vector norms) as opposed to the "ball monitoring" idea suggested in (Sharfman et al. [2007]) that sometimes fails for the distance provided by the l_1 norm, i.e. $\text{conv}\{\mathbf{v}(t_i), \mathbf{u}_1(t), \mathbf{u}_2(t), ...\mathbf{u}_n(t)\} \not\subseteq \cup B_1(\mathbf{v}(t_i), \mathbf{u}_j(t))$. Indeed, when $B_1(\mathbf{v}, \mathbf{u})$ is an l_1 ball of radius $\frac{1}{2}||\mathbf{v} - \mathbf{u}||_1$ centered at $\frac{\mathbf{v}+\mathbf{u}}{2}$, for example, $\mathbf{v} = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix}, \mathbf{u}_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \mathbf{u}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ (see Figure 1.2.3) one has $\text{conv}\{\mathbf{v}, \mathbf{u}_1, \mathbf{u}_2\} \not\subseteq B_1(\mathbf{v}, \mathbf{u}_1) \cup B_1(\mathbf{v}, \mathbf{u}_2)$.

In the next section we analyze the numerical experiments conducted in (Kogan [2012]) with the dataset described in Section 1 Chapter 4.

Algorithm 1

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 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 its local constraint δ
     the root sets $i = i + 1$
     go to Step 1.

- Stop

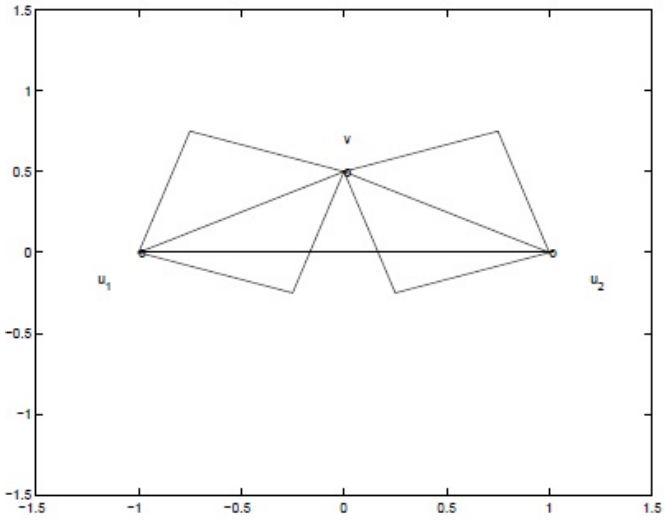


Fig. 1.2.3: failed cover by l_1 balls

1.3 Motivation and Contribution

Our goal is to monitor data streams with as little communication as possible over a sequence of discrete time instances that we shall denote by t. The time instances that

require communication between nodes are denoted by t_i, $i = 1, 2, ...$

The numerical experiments conducted with l_1, l_2 and l_∞ norm in (Kogan [2012]) with the dataset described in Section 1 Chapter 4 show that:

1. The number of time instances the mean violates (1.0.1) 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.

Table 1.3.1 presents a result of an experiment reported in (Kogan [2012]) for a distributed system with 10 nodes. The first row of the table shows the number of nodes simultaneously violating local constraint, the second one reports the number of time instances over the life time of the system when the violation of local constraint happened exactly at k nodes. The table reports total of 4006 time instances when the local constraint is violated. In 3034 out of 4006 time instances, communications with the root are triggered by constraint violations at exactly one node, in 620 time instances the violation was caused by exactly two nodes, and at no time instance violation was caused by all 10 nodes. This observation naturally leads to the idea of clustering nodes to further reduce communication load, and independent monitoring of the node clusters equipped with a coordinator.

# 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

Tab. 1.3.1: Number of local constraint violations simultaneously by k nodes, $r = 0.0025$, l_2 norm, the feature is "bosnia"

Clustering in general is a difficult problem, and many clustering problems are known to be NP-complete (no fast solution to them is known) (Brucker [1978]).That is,

the time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows. In this thesis we advance clustering approach to monitoring. The main contribution of this work is twofold:

1. We suggest to cluster nodes in order to reduce communication required, apply 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.

1.4 Overview of Dissertation

This thesis is organized as follows. In Chapter 2 we present a number of well known clustering algorithms and indicate how those can be used for monitoring data streams. However, observations show that a straightforward application of common clustering methods are not applicable to our problem. In Chapter 3 we present a specific strategy of monitoring distributed data streams through node clustering that yields a reduction in communication load. This novel type of clustering dictated by the problem at hand requires development of new algorithms and this thesis is a step in this direction. Experimental results on real-world data of monitoring with and without clustering as well as their comparison presented in Chapter 4. Using different norms we can see applications of monitoring data streams that benefit from clustering. Clustering, however does not offer a universal remedy. In Chapter 5 we indicate new research directions and conclude this thesis. Appendices summarize some useful properties of the first and second moments as well as detail the accounting of message transmission.

Chapter 2

Conventional Clustering Algorithms

Clustering or grouping document collections into conceptually meaningful clusters is a well-studied problem. As we described in Section 1 Chapter 1 the basic idea of creating a vector space model is first to extract unique content-bearing words from the set of documents treating these words as features and then represent each document as a vector of certain weighted word frequencies in this feature space (Salton et al. [1975]). Typically the document vectors are very high-dimensional. In addition, a single document typically contains only a small fraction of the total number of words in the entire collection; hence, the document vectors are generally very sparse, i.e., contain a lot of zero entries.

The k-means algorithm is a popular method for clustering a set of data vectors. An application of k-means though requires an initial partition of the data. While a number of initialization methods for the k-means clustering algorithm are available in the literature (Celebi et al. [2013]) we focus on PDDP which is briefly described next.

2.1 PDDP

A good partitioning of a vector set into a number of subsets is a difficult problem even in the case when the required number of subsets is only two. There is, however, an exception. When the dimension of the vector space is one, i.e. one has to deal with a scalar set, the problem is relatively easy. While real-life data is rarely one dimensional, a least squares one dimensional approximation can be constructed and used to cluster a multidimensional vector set. The Principal Direction Divisive Partitioning algorithm

briefly recalled below does just that. In the reminder of this section we denote by $||\mathbf{a}||$ the l_2 norm of a vector $\mathbf{a}$.

For a vector $\mathbf{a}$ and a line $\mathbf{l}$ in $\mathbf{R}^d$ denote by $\mathbf{P_l}(\mathbf{a})$ the orthogonal projection of $\mathbf{a}$ on $\mathbf{l}$. For a set of vectors $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ denote by $\mathbf{P_l}(\mathcal{A})$ the set of projections $\{\mathbf{P_l}(\mathbf{a}_1), \ldots, \mathbf{P_l}(\mathbf{a}_m)\}$. For a fixed vector set A the quantity

$$\sum_{i=1}^{m} ||\mathbf{a}_i - \mathbf{P_l}(\mathbf{a}_i)||^2$$

depends on the line $\mathbf{l}$. A line that minimizes this quantity (and provides the best least squares fit for the set A) defines a principal direction. This line passes through the arithmetic mean $\mu = \mu(\mathcal{A})$ of the vector set $\mathcal{A}$, and its direction vector is an eigenvector of the matrix BB^T that corresponds to the maximal eigenvalue. Here $B = [\mathbf{a}_1, \ldots, \mathbf{a}_m] - \mu\mathbf{e}^T$, and $\mathbf{e}$ is a vector of ones (for details see Boley [1998]).

A basic step of the Principal Direction Divisive Partitioning algorithm (PDDP) is the following:

1. Given a set of vectors $\mathcal{A}$ in $\mathbf{R}^d$ determine the one dimensional line $\mathbf{l}$ that provides the "best" approximation to $\mathcal{A}$.

2. Project $\mathcal{A}$ onto $\mathbf{l}$, and denote the projection of the set $\mathcal{A}$ by $\mathbf{P}$ (note that $\mathbf{P}$ is just a set of scalars). Denote the projection of a vector $\mathbf{a}$ by p.

3. Partition $\mathbf{P}$ into two subsets $\mathbf{P}_1$ and $\mathbf{P}_2$.

4. Generate the induced partition $\{\mathcal{A}_1, \mathcal{A}_2\}$ of $\mathcal{A}$ as follows:

$$\mathcal{A}_1 = \{\mathbf{a} \ : \ p \in \mathbf{P}_1\}, \text{ and } \mathcal{A}_2 = \{\mathbf{a} \ : \ p \in \mathbf{P}_2\} \tag{2.1.1}$$

The algorithm divides the entire collection into two clusters by using the principal direction. Each of these two clusters will be divided into two sub-clusters using the same

process recursively. The subdivision of a cluster is stopped when the cluster satisfies a certain "quality" criterion (such as, for example, cluster size, number of clusters, or cluster quality).

Implementation of the algorithm requires computation of the largest eigenvalue of the symmetric matrix BB^T. In many cases this task may not be performed analytically. While in the text mining application described in Section 1 Chapter 1 the space dimension $d = 4$ one of the coordinates is an affine function of three others. We now consider the case when each d dimensional data vector $\mathbf{a}$ can be written as

$$\mathbf{a} = \begin{bmatrix} \mathbf{b} \\ C\mathbf{b} + \mathbf{d} \end{bmatrix},$$

where $\mathbf{b} \in \mathbf{R}^{d_1}$, $\mathbf{d} \in \mathbf{R}^{d_2}$, $d_1 + d_2 = d$, and C is an $d_2 \times d_1$ matrix. For a vector set $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$ one has $\mu(\mathcal{A}) = \begin{bmatrix} \mu(\mathcal{B}) \\ C\mu(\mathcal{B}) + \mathbf{d} \end{bmatrix}$. We now turn to the matrix BB^T. Denoting $\mathbf{b}_i - \mu(\mathcal{B})$ by $\mathbf{v}_i$ we obtain

$$B = \begin{bmatrix} \mathbf{v}_1 \ldots \mathbf{v}_m \\ C\mathbf{v}_1 \ldots C\mathbf{v}_m \end{bmatrix} = \begin{bmatrix} I \\ C \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \ldots \mathbf{v}_m \end{bmatrix}, \text{ where } I \text{ is the } d_1 \times d_1 \text{ identity matrix.}$$

Since rank $B \leq d_1$ one has rank $BB^T \leq d_1$, and the number of nonzero eigenvalues of BB^T does not exceed d_1. For our text mining application $d_1 = 3$, and the nonzero eigenvalues can be obtained by solving a cubic equation, i.e., the eigenvector for BB^T corresponding to the largest eigenvalue can be obtained just by solving a system of linear equations.

PDDP by itself generates good clustering results. Those could be further improved by applying $k-$means clustering to partitions generated by PDDP (see e.g. Boley [1998]). Next we briefly recall a number of versions of $k-$means.

2.2 Batch k-means

Batch $k-$means is by far the most popular clustering algorithm. The algorithm is scalable, and easy to implement. The algorithm is centered around the concept of "centroid"–the best vector representative for a vector set introduced first by Steinhaus in 1956 (see Steinhaus [1956], MacQueen et al. [1967], Diday [1973], Lloyd [1982] and Forgy [1965]).

For a set of vectors $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\} \subset \mathbf{R}^n$, and a "distance" function $d(\mathbf{x}, \mathbf{a})$ define a centroid $\mathbf{c} = \mathbf{c}(\mathcal{A})$ of the set $\mathcal{A}$ as a solution of the minimization problem

$$\mathbf{c} = \arg\min \left\{ \sum_{\mathbf{a} \in \mathcal{A}} d(\mathbf{x}, \mathbf{a}),\ \mathbf{x} \in \mathcal{C} \right\}, \tag{2.2.1}$$

where $\mathcal{C} \subset \mathbf{R}^n$.

We call d a "distance-like" function because even in the classical implementation of $k-$means $d(\mathbf{x}, \mathbf{a}) = \|\mathbf{x} - \mathbf{a}\|_2^2$, the square of the l_2 norm, that fails to be a distance function (the triangle inequality does not hold). Further, $k-$ means works with a wide class of functions called Bregman divergences and failing to be distances (Kullback–Leibler divergence is one of them, see Banerjee et al. [2005]).

The quality of the set $\mathcal{A}$ is denoted by $q(\mathcal{A})$ and is defined by

$$q(\mathcal{A}) = \sum_{\mathbf{a} \in \mathcal{A}} d(\mathbf{c}, \mathbf{a}), \quad \text{where } \mathbf{c} = \mathbf{c}(\mathcal{A}) \tag{2.2.2}$$

(we set $q(\emptyset) = 0$ for convenience). Let $\Pi = \{\pi_1, \ldots, \pi_k\}$ be a partition of $\mathcal{A}$, i.e.

$$\bigcup_i \pi_i = \mathcal{A}, \text{ and } \pi_i \cap \pi_j = \emptyset \text{ if } i \neq j.$$

We define the quality of the partition Π by

$$Q(\Pi) = q(\pi_1) + \cdots + q(\pi_k). \tag{2.2.3}$$

We aim to find a $k-$ cluster partition $\Pi^{\min} = \{\pi_1^{\min}, \ldots, \pi_k^{\min}\}$ that *minimizes* the value of the objective function Q. The problem is known to be NP–hard (Brucker [1978]), and we are looking for algorithms that generate "reasonable" solutions. It is easy to see that centroids and partitions are associated as follows:

1. Given a partition $\Pi = \{\pi_1, \ldots, \pi_k\}$ of the set $\mathcal{A}$ one can define the corresponding centroids $\{\mathbf{c}(\pi_1), \ldots, \mathbf{c}(\pi_k)\}$ by:

$$\mathbf{c}(\pi_i) = \arg\min \left\{ \sum_{\mathbf{a} \in \pi_i} d(\mathbf{x}, \mathbf{a}),\ \mathbf{x} \in \mathcal{C} \right\}, \tag{2.2.4}$$

   where $\mathcal{C}$ is a predefined subset of $\mathbf{R}^d$.

2. For a set of k "centroids" $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ one can define a partition $\Pi = \{\pi_1, \ldots, \pi_k\}$ of the set $\mathcal{A}$ by:

$$\pi_i = \{\mathbf{a} \ :\ \mathbf{a} \in \mathcal{A},\ \ d(\mathbf{c}_i, \mathbf{a}) \leq d(\mathbf{c}_l, \mathbf{a}) \text{ for each } l = 1, \ldots, k\} \tag{2.2.5}$$

   (we break ties arbitrarily). Note that, in general, $\mathbf{c}(\pi_i) \neq \mathbf{c}_i$.

The classical batch k–means algorithm is a procedure that iterates between the two steps described above to generate a partition Π' from a partition Π.

While $0 \leq Q(\Pi') \leq Q(\Pi)$ and the process described above converges, it rarely converges to the global minimum. In case we start with an arbitrary poor clustering the batch $k-$ means algorithm returns a poor clustering as the final ouput. Take the simple scalar case $\mathcal{A} = \{0, 2, 3\}$, and the initial partition $\Pi^{(0)} = \left\{\pi_1^{(0)}, \pi_2^{(0)}\right\}$ where $\pi_1^{(0)} = \{0, 2\}$, and $\pi_2^{(0)} = \{3\}$ an application of batch $k-$means to $\Pi^{(0)}$ does not change

the partition, and misses a better partition $\Pi^{(1)} = \left\{\pi_1^{(1)}, \pi_2^{(1)}\right\}$ with $\pi_1^{(1)} = \{0\}$, and $\pi_2^{(1)} = \{2, 3\}$. The reason for this phenomenon along with a possible remedy is suggested in (Dhillon et al. [2002]). Before the relevant material is briefly recalled in the next section we remark that an application of incremental clustering described below to the scalar dataset $\mathcal{A}$ generates partition $\Pi^{(1)}$. This observation suggests to use PDDP to generate initial partitions to $k-$means like algorithms.

2.3 Incremental k-means

The failure of batch $k-$means to discover a better partition $\Pi^{(1)}$ stems from a simple fact that Step 2 of the procedure ignores change of centroids due to data-vectors' movement governed by (2.2.5). A way to accurately account for the centroid change is to allow a single data-vector movement during one iteration of the algorithm. This version of $k-$means is described, for example, in the classical manuscript (Duda et al. [2012]).

While more accurate, incremental $k-$means changes cluster affiliation of only one vector per iteration. As compared to batch $k-$means the algorithm requires many more iterations to converge, hence is time consuming. One way to enhance incremental $k-$means is by expanding the local search to seek a chain of moves in a Kernighan-Lin fashion instead of just one more. This enhancement leads to a better local maximum (Dhillon et al. [2002]).

We take again the simple scalar case described in Section 2 Chapter 2 and apply incremental $k-$means. Then we will get the partition $\Pi^{(1)}$ which is a better partition than $\Pi^{(0)}$ since

$$0 \leq Q(\Pi^{(1)}) \leq Q(\Pi^{(0)})$$

Incremental $k-$means becomes effective in the case of small clusters (cluster of size 100 or less) (Dhillon et al. [2002]). We next discuss a "merger" of two algorithms in order to achieve better results.

2.4 Batch k-means followed by incremental k-means

While more accurate incremental $k-$means is not as fast as the batch algorithm. To benefit from speed of the batch algorithm and accuracy of the incremental $k-$means a number of contributions suggested to "merge" both algorithms as follows:

1. run batch $k-$means until it stops.

2. run one iteration of incremental $k-$means

3. if the iteration incremental $k-$means changed the partition

   go to Step 1

   else

   Stop.

All numerical computations associated with Step 2 of the algorithm have been already performed in Step 1. The improvement over batch $k-$means comes, therefore, at virtually no additional computational expense (Dhillon et al. [2004]). The possibility of the "merger" was first indicated, perhaps, in (Spath [1980]), and formally introduced in ([1985]). Later the "merger" was independently rediscovered by many other authors[1]. Sequential application

$$\text{PDDP} \longrightarrow \text{batch } k-\text{means} \longrightarrow \text{incremental } k-\text{means}$$

generates good tight clusters. Next we describe how these tight clusters can be used for node clustering.

[1] confirming the old adage that "success has many parents while failure is an orphan."

2.5 Node Clustering with Classical Clustering Algorithms

To simplify the exposition we first consider a two cluster $\{\pi_1, \pi_2\}$ partition problem for a given set of n vectors $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\} \subset \mathbf{R}^d$. We are seeking a partition $\Pi = \{\pi_1, \pi_2\}$ so that

$$\mathcal{A} = \pi_1 \bigcup \pi_2, \ \pi_1 \bigcap \pi_2 = \emptyset$$

and the partition Π quality $Q(\Pi)$ given by

$$Q(\Pi) = \max \left\{ \left\| \frac{1}{|\pi_1|} \sum_{\mathbf{a} \in \pi_1} \mathbf{a} \right\|, \left\| \frac{1}{|\pi_2|} \sum_{\mathbf{a} \in \pi_2} \mathbf{a} \right\| \right\}$$

is minimized. We denote the size of π_i by $|\pi_i|$. Due to convexity of any norm one has

$$\begin{aligned} \left\| \frac{1}{|\mathcal{A}|} \sum_{\mathbf{a} \in \mathcal{A}} \mathbf{a} \right\| &= \left\| \frac{|\pi_1|}{|\mathcal{A}|} \frac{1}{|\pi_1|} \sum_{\mathbf{a} \in \pi_1} \mathbf{a} + \frac{|\pi_2|}{|\mathcal{A}|} \frac{1}{|\pi_2|} \sum_{\mathbf{a} \in \pi_2} \mathbf{a} \right\| \\ &\leq \frac{|\pi_1|}{|\mathcal{A}|} \left\| \frac{1}{|\pi_1|} \sum_{\mathbf{a} \in \pi_1} \mathbf{a} \right\| + \frac{|\pi_2|}{|\mathcal{A}|} \left\| \frac{1}{|\pi_2|} \sum_{\mathbf{a} \in \pi_2} \mathbf{a} \right\| \\ &\leq \frac{|\pi_1|}{|\mathcal{A}|} Q(\Pi) + \frac{|\pi_2|}{|\mathcal{A}|} Q(\Pi) = Q(\Pi). \end{aligned}$$

This inequality shows that the norm of the mean is a lower bound for $Q(\Pi)$. Take the simple scalar case $\mathcal{A} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 15\}$ (Semenov [2014]), where the mean of $\mathcal{A}$ is 6. As we saw above $6 \leq Q(\Pi)$. Again, we are seeking for a two cluster partition that minimizes

$$Q(\Pi) = \max \left\{ \left\| \frac{1}{|\pi_1|} \sum_{\mathbf{a} \in \pi_1} \mathbf{a} \right\|, \left\| \frac{1}{|\pi_2|} \sum_{\mathbf{a} \in \pi_2} \mathbf{a} \right\| \right\}.$$

Since $Q(\Pi)$ cannot be less than 6 a best cluster partition would be $\Pi^{(0)} = \left\{ \pi_1^{(0)}, \pi_2^{(0)} \right\}$, where $\pi_1^{(0)} = \{6\}$ and $\pi_2^{(0)} = \{1, 2, 3, 4, 5, 7, 8, 9, 15\}$. Hence $Q(\Pi) = 6$. This example is interesting because one can easily build a 3 cluster optimal partition $\pi_1^{(0)} = \{6\}$, $\pi_2^{(0)} = \{5, 7\}$, $\pi_3^{(0)} = \{1, 2, 3, 4, 8, 9, 15\}$, a 4 cluster optimal partition $\pi_1^{(0)} = \{6\}$, $\pi_2^{(0)} = \{5, 7\}$,

$\pi_3^{(0)} = \{4, 8\}$, $\pi_4^{(0)} = \{1, 2, 3, 9, 15\}$, etc. We next show how to build an optimal partition for a special particular case of the data set.

Assume that $n = 2m$, and the vector set $\mathcal{A}$ consists of two identical copies of m vectors, i.e.

$$\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathbf{a}_1, \ldots, \mathbf{a}_m\}.$$

If $\pi_1^o = \pi_2^o = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$, then $|\pi_1^o| = |\pi_2^o| = \frac{1}{2}|\mathcal{A}|$, and

$$\max\left\{\frac{1}{|\pi_1^o|}\left\|\sum_{\mathbf{a}\in\pi_1^o}\mathbf{a}\right\|, \frac{1}{|\pi_2^o|}\left\|\sum_{\mathbf{a}\in\pi_2^o}\mathbf{a}\right\|\right\} = \left\|\frac{1}{|\mathcal{A}|}\sum_{\mathbf{a}\in\mathcal{A}}\mathbf{a}\right\|,$$

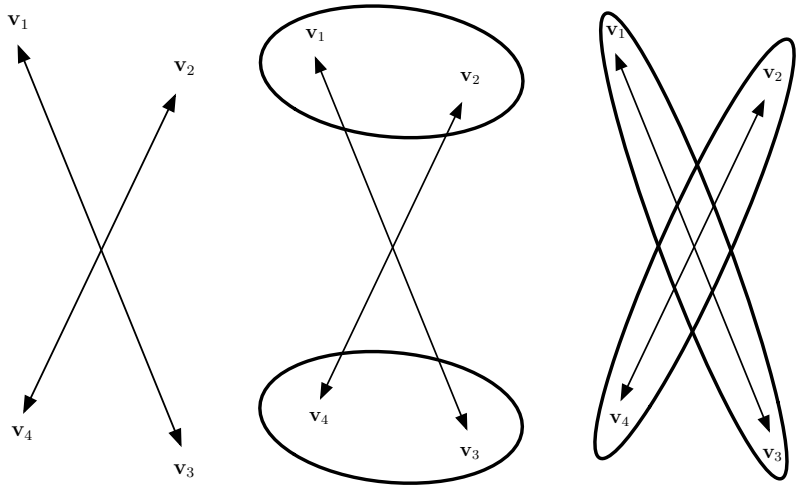
i.e., $\{\pi_1^o, \pi_2^o\}$ is an optimal partition.

This observation motivates the following two cluster $\Pi = \{\pi_1, \pi_2\}$ partition strategy:

1. Apply any clustering algorithm to the dataset $\mathcal{A}$ to generate clusters of size 2.
2. Select one vector from each cluster generated and assign selected vectors to cluster π_1.
3. Assign remaining vectors to cluster π_2.

Generalization of this strategy to a k cluster partition and full description of the algorithm is beyond the scope of our work and will be provided elsewhere.

Clustering in general is a difficult problem and there does not exist a best clustering method that is, one which is superior to the other methods. For this reason there are many clustering problems that are known to be NP-complete (Brucker [1978]). Unlike classical clustering approaches that attempt to collect together similar vectors (Mirkin [2012]), we are looking for clusters with dissimilar vectors 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 for our problem. To illustrate this, assume that we have 4 vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ and $\mathbf{v}_4$ as shown in Figure 2.5.1a. Note that their mean is zero. The classical clustering



(a) *(b)* *(c)*

Fig. 2.5.1

approach suggests to cluster similar vectors together but in our problem this leads to bad sub-clusters as shown in Figure 2.5.1b. On the other hand, when we cluster together dissimilar vectors (Figure 2.5.1c) we can see that the mean of the clusters is minimized.

Since our idea is based on minimization of the total average change within a cluster, common clustering methods are not applicable to our problem. In Chapter 3 we present a specific strategy for monitoring distributed data streams through node clustering that yields a reduction in communication load.

Chapter 3

Adaptive Clustering for Monitoring Distributed Data Streams

A standard clustering problem is often described as finding and describing cohesive or homogeneous chunks in data, the clusters (see e.g. Mirkin [2012]). Motivated by the results in (Kogan [2012]), the authors in (Kogan and Malinovsky [2013]) present an approach based on convex analysis techniques and clustering. Their idea is to apply a different clustering strategy attempting to balance out vectors assigned to the same cluster, and by doing so to minimize the norm of the clusters average. Thus by clustering together the "longest" vector $\mathbf{v}_{n_L}(t) - \mathbf{v}_{n_L}(t_i)$ with the "shortest" $\mathbf{v}_{n_S}(t) - \mathbf{v}_{n_S}(t_i)$ reduction of the communication load is achieved. This idea may result in the mean of the two vector cluster being shorter than δ (as defined in Chapter 1). To illustrate this further 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. Assume now that the local constraint is violated at n_1 at time t, 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. Thus 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. Therefore, node clustering may lead to communication savings. It is shown in (Kogan and Malinovsky [2013]) that clustering together just two nodes may reduce communication by about 10%.

Since dissimilarity of vectors in each cluster should be maximized (and the similarity should be minimized) a direct application of center-based clustering algorithms (such as, for example, $k-$means) is not possible. Next, we extend the idea of clustering together dissimilar vectors as suggested in (Kogan and Malinovsky [2013]) to the general case-invloving many nodes, arbitrary functions, and high-dimensional data.

3.1 Mathematical Formulation

Motivated by the results of Table 1.3.1 and (Kogan and Malinovsky [2013]) we advance the node clustering approach and demonstrate additional computation savings (see Barouti et al. [2014a], Barouti et al. [2014b]). For the problem at hand we have a set of nodes $\mathbf{N}$ and we would like to partition it into k disjoint clusters $\Pi = \{\pi_1, \ldots, \pi_k\}$ so that

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

Then each cluster π_i will be equipped with a "coordinator" $\mathbf{c}_i$ (one of the clusters' node) and

- if $\mathbf{n}' \in \pi_i$ violates its local constraint at time $t > t_j$, where t_j was the last time the violation took place, then $\mathbf{c}_i$ collects vectors $\mathbf{v}_n(t) - \mathbf{v}_n(t_j)$ from all $\mathbf{n} \in \pi_i$.

- $\mathbf{c}_i$ computes the mean of π_i and checks if the total change within cluster π_i does not

exceed δ, i.e.,

$$\frac{1}{|\pi_i|}\left|\left|\sum_{\mathbf{n}\in\pi_i}[\mathbf{v}_n(t)-\mathbf{v}_n(t_j)]\right|\right|<\delta. \tag{3.1.1}$$

If (3.1.1) holds for each cluster then

$$\left|\left|\sum_{\mathbf{n}\in\mathbf{N}}\frac{\mathbf{v}_n(t)-\mathbf{v}_n(t_j)}{n}\right|\right|\leq\sum_{i=1}^{k}\frac{|\pi_i|}{n}\left|\left|\sum_{\mathbf{n}\in\pi_i}\frac{\mathbf{v}_n(t)-\mathbf{v}_n(t_j)}{|\pi_i|}\right|\right|<\delta. \tag{3.1.2}$$

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 "old" mean $\frac{1}{n}\sum_{\mathbf{n}\in\mathbf{N}}\mathbf{v}_\mathbf{n}(t_j)$ belongs to this set and recomputation of $\mathbf{v}(t)$ is not needed. If (3.1.1) fails for at least one cluster, then the cluster coordinator alerts the root (a node of the set $\mathbf{N}$), and the mean of the entire dataset $\mathbf{v}(t)$ is recomputed by the root.

By this way we can check if (1.0.1) holds. To conclude we can say that our aim is to identify k and a k cluster partition Π^o that **minimizes**

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

for $t>t_j$.

The monitoring problem requires to assign nodes $\{\mathbf{n}_{i_1},\ldots,\mathbf{n}_{i_k}\}$ to the same cluster π so that the total average change within cluster π

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

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. Thus we will achieve reduction of the communication load (see Chapter 4).

It is important to see that the proposed partition quality $Q(\Pi)$ 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 the next section).

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 (3.1.1) is proportional to the size of a cluster, the individual clusters' sizes should affect the clustering quality $q(\pi)$.

In the next section we address these problems.

3.2 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

$$q(\pi) = \frac{1}{|\pi|} \left|\left| \sum_{\mathbf{n} \in \pi} [\mathbf{v_n}(t) - \mathbf{v_n}(t_j)] \right|\right| + \alpha|\pi|, \tag{3.2.1}$$

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

$$Q(\Pi) = \max_{i \in \{1,\ldots,k\}} q(\pi_i), \tag{3.2.2}$$

When $\alpha = 0$ the partition that minimizes $Q(\Pi)$ is a single cluster partition (that we would like to avoid). Larger values of α force clusters' size to decrease.
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 chapter we shall focus on

$$0 < \alpha < \max_{\mathbf{n} \in \mathbf{N}} ||\mathbf{v_n}(t) - \mathbf{v_n}(t_j)|| \,. \tag{3.2.3}$$

The choice of 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 behavior 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:

- $\mathbf{h_n}(t_0) = 0$
- $\mathbf{h_n}(t_1) = \mathbf{v_n}(t_1) - \mathbf{v_n}(t_0)$
- $\mathbf{h_n}(t_2) = \mathbf{v_n}(t_2) - \mathbf{v_n}(t_1) + \frac{1}{2}[\mathbf{v_n}(t_1) - \mathbf{v_n}(t_0)]$
- ...
- $\mathbf{h_n}(t_j) = \mathbf{v_n}(t_j) - \mathbf{v_n}(t_{j-1}) + \frac{1}{2}[\mathbf{v_n}(t_{j-1}) - \mathbf{v_n}(t_{j-2})] + ... + \frac{1}{2^{j-1}}[\mathbf{v_n}(t_1) - \mathbf{v_n}(t_0)]$

The vectors $\mathbf{h_n}(t_j)$ accumulate the history of changes, with older changes assigned smaller weights. We shall use the vectors $\{\mathbf{h_n}(t_j)\}$ to generate a node partition at time t_j. We note that scaling of the vector set that should be clustered does not change the induced optimal partitioning of the nodes. When the vector set is scaled by the magnitude of the longest vector in the set, the range for α conveniently shrinks to $[0, 1]$. In what follows we set $h = \max_{\mathbf{n} \in \mathbf{N}} ||\mathbf{h_n}(t_j)||$, 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). We set $k = n$ and loop the following procedure until the number of clusters reduces to $k = 2$.

Algorithm 2

Incremental Clustering

- Set $k = n$.
- do until $k > 2$:
  1. in partition Π^k identify the cluster π_j of maximal quality , i.e.,

     $$q(\pi_j) \geq q(\pi_i),\ i \neq j,$$

     or

     $$\frac{1}{|\pi_j|}\left|\left|\sum_{\mathbf{a}\in\pi_j}\mathbf{a}\right|\right| + \alpha|\pi_j| \geq \frac{1}{|\pi_i|}\left|\left|\sum_{\mathbf{a}\in\pi_i}\mathbf{a}\right|\right| + \alpha|\pi_i|,\ i \neq j.$$

  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 in step 1.
  3. Build partition Π^{k-1} by merging clusters π_j and π_i.
  4. Set $k = k - 1$.
  5. go to Step 1.
- Stop.

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 (see step 2 of monitoring algorithm with clustering below). Taking into account the distribution of the data streams at each node can further reduce communication. We illustrate this statement by a sim-

ple example involving two nodes. If, for example, there is a reason to believe that the inequality

$$2\|\mathbf{v}_1(t) - \mathbf{v}_1(t_i)\| \leq \|\mathbf{v}_2(t) - \mathbf{v}_2(t_i)\| \tag{3.2.4}$$

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 (3.2.4). With no additional assumptions about the stream data distribution this information is not available. Unlike (Keren et al. [2012]) we refrain from making assumptions regarding the underlying data distributions; instead, we estimate the weights through past values $\mathbf{v}_\mathbf{n}(t)$, $\mathbf{n} \in \mathbf{N}$ as explained below.

In what follows we denote the arithmetic mean $\frac{\mathbf{x}_1 + \cdots + \mathbf{x}_m}{m}$ of a vector set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbf{R}^d$ by $\mu(\mathbf{X})$. With slight abuse of notations the central second moment $\sum_{i=1}^m (\mathbf{x}_i - \mu(\mathbf{X}))^T (\mathbf{x}_i - \mu(\mathbf{X}))$ is denoted by $\sigma^2(\mathbf{X})$. At the initial time t_0 all nodes report their vectors $\mathbf{v}_\mathbf{n}(t_0)$ to the root, the root computes the average, and the the distance $\delta(\mathbf{r})$ from the average to the boundary of $\mathbf{Z}_+(f)$. At this point we define $\delta(\mathbf{n}) = \delta(\mathbf{r})$, for each $\mathbf{n} \in \mathbf{N}$.

We now focus on a particular node $\mathbf{n}$. Consider first m time instances $t^1, t^2, \ldots, t^m$ and the vector set

$$\mathbf{V}'_\mathbf{n} = \{\mathbf{v}'_\mathbf{n}(t^1), \ldots, \mathbf{v}'_\mathbf{n}(t^m)\},$$

where

$$\mathbf{v}'_{\mathbf{n}}(t^m) = \mathbf{v}_{\mathbf{n}}(t^m),\ \mathbf{v}'_{\mathbf{n}}(t^{m-1}) = \frac{1}{2}\mathbf{v}_{\mathbf{n}}(t^{m-1}),\ \ldots, \mathbf{v}'_{\mathbf{n}}(t^1) = \frac{1}{2^{m-1}}\mathbf{v}_{\mathbf{n}}(t^1).$$

The node constraint $\delta(\mathbf{n})$ introduced below depends on the arithmetic mean $\mu(\mathbf{V}'_{\mathbf{n}})$ and the central second moment

$$\sigma^2(\mathbf{V}'_{\mathbf{n}}) = \sum_{i=0}^{m} \left(\mathbf{v}'_{\mathbf{n}}(t_i) - \mu(\mathbf{V}'_{\mathbf{n}})\right)^T \left(\mathbf{v}'_{\mathbf{n}}(t_i) - \mu(\mathbf{V}'_{\mathbf{n}})\right)$$

of the node $\mathbf{n}$. We denote $\mu(\mathbf{V}'_{\mathbf{n}})$ by $\mu_{\mathbf{n}}$, and $\sigma^2(\mathbf{V}'_{\mathbf{n}})$ by $\sigma^2_{\mathbf{n}}$. Since $||\mathbf{v}'_{\mathbf{n}}(t^i) - \mu_{\mathbf{n}}|| \leq \sqrt{\frac{m-1}{m}\sigma^2_{\mathbf{n}}}$, $i = 1, \ldots, m$ (see Appendix 1) we define

$$W_{\mathbf{n}}(t^m) = W_{\mathbf{n}} = ||\mu_{\mathbf{n}}|| + \sqrt{\frac{m-1}{m}\sigma^2_{\mathbf{n}}}. \tag{3.2.5}$$

We note that although the bound $\sqrt{\frac{m-1}{m}\sigma^2_{\mathbf{n}}}$ may be very conservative, the same conservative criterion is applied uniformly to every node.

If at time t^m the root constraint $\delta(\mathbf{r})$ is updated, each node $\mathbf{n}$ broadcasts $W_{\mathbf{n}}(t^m) = W_{\mathbf{n}}$ to the root, the root computes $W = \sum_{\mathbf{n}\in\mathbf{N}} W_{\mathbf{n}}$, and transmits the updated $\delta(\mathbf{n}) = w_{\mathbf{n}}\delta(\mathbf{r})$ where $w_{\mathbf{n}} = n \times \frac{W_{\mathbf{n}}}{W}$ (so that $\sum_{\mathbf{n}\in\mathbf{N}} w_{\mathbf{n}} = n$) back to node $\mathbf{n}$. For a coordinator $\mathbf{c}$ of a node cluster π the constraint $\delta(\mathbf{c}) = \frac{1}{|\pi|}\sum_{\mathbf{n}\in\pi}\delta(\mathbf{n})$.

Node constraints $\delta(\mathbf{n})$ based on the first moment only are introduced in (Barouti et al. [2014b]). In Chapter 4 we provide monitoring results for node constraints based on the first and second moments, and compare the results with those reported in (Barouti et al. [2014a]) as well as monitoring with no clustering reported in (Kogan [2012]).

Algorithm 3

Monitoring Threshold Function with Clustering

- 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 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. set $t = t_i$
     do
     set violation$= 0,\ t = t + 1$
     for each $\mathbf{n} \in \mathbf{N}$
     If $||\mathbf{v_n}(t) - \mathbf{v_n}(t_i)|| \geq \delta$
     violation++
     endif
     while (violation=0)
  3. set $i = i + 1$, and $t_i = t$
  4. violator node $\mathbf{n}$ notifies the root about the violation of its local constraint δ
  5. The root requests vectors $\mathbf{v_n}(t_i)$ and weights $W_\mathbf{n}(t_i)$.
     The root forms a partition $\Pi = \{\pi_1, \ldots, \pi_k\}$ (based on incremental clustering algorithm) and sends node and coordinator constraints $\delta(\mathbf{n})$ and $\delta(\mathbf{c})$ to nodes and coordinators.
  6. do for each $\pi \in \Pi$
     do for each $\mathbf{n} \in \pi$
     If $\delta(\mathbf{n}) \leq ||\mathbf{v_n}(t) - \mathbf{v_n}(t_i)||$
     If $\delta(\mathbf{c}) \leq \frac{1}{|\pi|}\left|\left|\sum_{\mathbf{n}\in\pi} \mathbf{v_n}(t) - \sum_{\mathbf{n}\in\pi} \mathbf{v_n}(t_i)\right|\right|$
     notify root about coordinator violation
     root updates the distance $\delta(\mathbf{r})$ and go to Step 3
     endif
     endif

- Stop

Chapter 4

Experimental Results

In this chapter we first describe our data and since the weights described in Chapter 3 depend on the arithmetic mean and the central second moment, we provide monitoring results for node constraints based on the first and second moments. Next we compare the results with those reported in (Kogan [2012]) (monitoring with no clustering).

4.1 Data

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 $\mathbf{v}_i$ in ascending order with respect to document ID, and selecting feature vectors for the stream in the round-robin fashion. The feature vectors are generated as described in text mining application (see Chapter 1 Section 1).

Each document in the Reuters Corpus RCV1–V2 is labeled as belonging to one or more categories. We label a document as "relevant" if it belongs to the "CORPORATE/INDUSTRIAL" ("CCAT") category, and "spam" otherwise. In the experiments our goal is to select features that are most relevant to the "CCAT" category. Thus by following (Sharfman et al. [2007]) we focus on three features: "bosnia", "ipo", and "febru" that display different characteristic behaviour. In each experiment we want to detect for each feature, at any given time, whether its information gain (1.1.2) is above or below a

given threshold value (i.e. (1.1.3)). 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 $\frac{714,265}{10}$ times. At any given time the information gain of a feature is based on the documents contained at the time in the sliding windows of all the nodes. Thus based on all the documents contained in the sliding window at each one of the $\frac{714,266}{10}$ time instances we compute and graph $\frac{714,266}{10}$ information gain values for the feature "bosnia" (see Figure 4.1.1). Thus the following figure shows us how much information each feature contains as the stream evolve. In Figure 4.1.1 we can observe that the information gain for the feature "bosnia" displays an oscillating trend as the stream evolve.

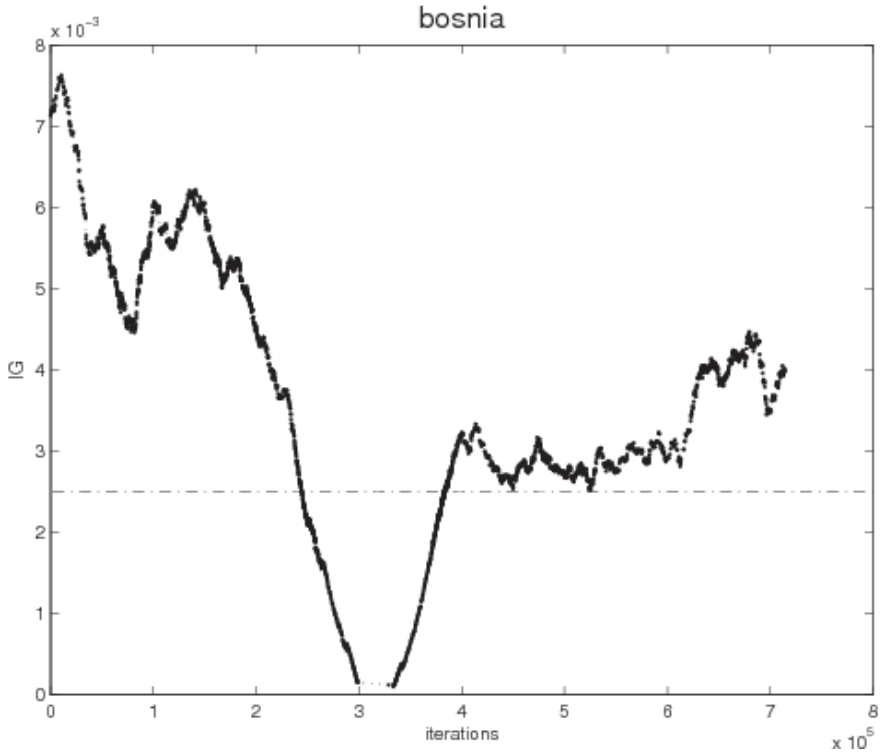


Fig. 4.1.1: information gain values for the feature "bosnia"

4.2 Monitoring with Incremental Clustering

For the experiments described below the goal is to monitor the inequality $f(\mathbf{v}) - r > 0$ while minimizing communication between the nodes. We assume that new data arrive simultaneously at each node.

Before we analyze the results of our clustering approach we can see some numerical results of the approach described in (Kogan [2012]) for a variety of vector norms. The results obtained for "febru" without clustering are presented in Table 4.2.1, for "ipo" in 4.2.2 and for "bosnia" in Table 4.2.3. It is easy to see that the smallest number of communications is obtained when l_1 norm is used.

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

Tab. 4.2.1: Number of mean computations, and broadcasts for feature "febru" with threshold $r = 0.0025$, no clustering

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

Tab. 4.2.2: Number of mean computations, and broadcasts for feature "ipo" with threshold $r = 0.0025$, no clustering

As we saw in Table 1.3.1, an application of the procedure decribed in (Kogan [2012]) to data streams generated from the Reuters Corpus RCV1-V2 leads to 3034 out of 4006 time instances, communications with the root are triggered by constraint violations at exactly one node. These results and the idea of clustering together dissimilar vectors suggested in (Kogan and Malinovsky [2013]) motivated us to run the node clustering monitoring presented in Chapter 3.

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

Tab. 4.2.3: Number of mean computations, and broadcasts, for feature "bosnia" with threshold $r = 0.0025$, no clustering

The previous work (Barouti et al. [2014b]) reported monitoring results obtained with the incremental clustering algorithm with weights $W_{\mathbf{n}} = \|\mu_{\mathbf{n}}\|$ only. We shall call this implementation of the algorithm "first moment incremental clustering" (FMIC). The clustering algorithm with weights $W_{\mathbf{n}} = ||\mu_{\mathbf{n}}|| + \sqrt{\frac{m-1}{m}\sigma^2_{\mathbf{n}}}$ (3.2.5) introduced in Chapter 3 will be referred to as the "second moment incremental clustering" (SMIC). Moreover, as we mentioned before the cluster's size affects the frequency of the updates of the mean as well as the communication load. For this reason we introduced the cluster quality $q(\pi) = \frac{1}{|\pi|}\left|\left|\sum_{\mathbf{n}\in\pi}[\mathbf{v}_{\mathbf{n}}(t) - \mathbf{v}_{\mathbf{n}}(t_j)]\right|\right| + \alpha|\pi|$ (3.2.1). In this section we report and compare results generated by the algorithms for the threshold $r = 0.0025$ and $\alpha = 0.05, 0.10, \ldots, 0.95$. By keeping α between 0.05 and 0.95 we force nodes to cluster.

The choice of α that generates best results obtained by an application of FMIC to the features "febru," "ipo" and "bosnia" are presented in Tables 4.2.4, 4.2.5 and 4.2.6. The best results for l_1, l_2, and l_∞ norms with respect to α are presented in Table 4.2.4 in case of "febru" where the coordinators' constraints are not violated, and the root mean updates are decreased significantly. The coordinator mean is referred only to non singleton clusters. In Table 4.2.4 the root mean is updated due to violations in singletons. Note that the root mean is updated only if there is a violation in a singleton cluster or there is a coordinator violation in a cluster. In case of a coordinator violation in a cluster the coordinator mean is updated as well. Comparing results of Table 4.2.4 to Table 4.2.1 we can see that the number of broadcasts (communication between the nodes) decreases by about 50%.

norm	α	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

Tab. 4.2.4: Number of root and coordinator mean computations, and total broadcasts for feature "febru" with threshold $r = 0.0025$ and the "first moment clustering"

Table 4.2.5 demonstrates significant inside cluster activity, and a decrease in root mean updates. Last we turn to the feature "bosnia". Application of clustering to monitoring this feature information gain appears to be far less successful (see Table 4.2.6). 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 4.2.6). In case of the l_1 norm, the number of broadcasts increases.

norm	α	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

Tab. 4.2.5: Number of root and coordinator mean computations, and total broadcasts for feature "ipo" with threshold $r = 0.0025$ and the "first moment clustering"

norm	α	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

Tab. 4.2.6: Number of root and coordinator mean computations, and total broadcasts for feature "bosnia" with threshold $r = 0.0025$ and the "first moment clustering"

The corresponding results with respect to α generated by SMIC to the features "febru," "ipo" and "bosnia" are provided in Tables 4.2.7, 4.2.8 and 4.2.9. Results in Table 4.2.7 show a significant decrease in the number of broadcasts as compared to results in Table 4.2.4. Next we turn to the features "ipo" and "bosnia" and report results with the lowest number of broadcasts. Application of SMIC leads to results provided in Table 4.2.8. The table demonstrates significant inside cluster activity, and a significant decrease in broadcasts due to the second moment. Finally we turn to the feature "bosnia." The second moment clustering further significantly reduces the number of broadcasts, see Table 4.2.9.

norm	best α	root mean update	coordinator mean update	total broadcasts
l_1	0.85	883	0	23859
l_2	0.75	833	0	22509
l_∞	0.75	854	0	23076

Tab. 4.2.7: Number of root and coordinator mean computations, and total broadcasts for feature "febru" with threshold $r = 0.0025$ and the "second moment clustering"

norm	best α	root mean update	coordinator mean update	total broadcasts
l_1	0.50	4585	121	127345
l_2	0.35	6304	421	180536
l_∞	0.30	8405	842	240455

Tab. 4.2.8: Number of root and coordinator mean computations, and total broadcasts for feature "ipo" with threshold $r = 0.0025$ and the "second moment clustering"

As the results show sometimes clustering leads to communicational savings, however clustering does not offer a universal remedy; in some cases better performance is achieved with no clustering. In the next Chapter we discuss possible future directions that may reduce the communication load.

norm	best α	root mean update	coordinator mean update	total broadcasts
l_1	0.65	1749	8	47717
l_2	0.75	1940	4	52510
l_∞	0.65	1756	8	47958

Tab. 4.2.9: Number of root and coordinator mean computations, and total broadcasts for feature "bosnia" with threshold $r = 0.0025$ and the "second moment clustering"

Chapter 5

Discussion

In this thesis we consider application of clustering to monitoring data streams in a distributed system. Unlike standard clustering algorithms that aiming at collections of similar data items into same clusters, monitoring requires clusters with dissimilar vectors canceling each other as much as possible. A straightforward application of a standard clustering algorithm is, therefore, not possible.

The clustering strategy suggested is based on minimization of a combination of the total average change within a cluster and the cluster size. The nodes are re–clustered each time the entire dataset mean is updated. Note that each computation of $\delta(r)$ (the distance from the mean of the entire set and the zero set of the function) triggers recomputations of node constraints $\delta(\mathbf{n})$ that carry past history. This study uses the first and second moments to recompute node constraints. A possible future direction would be to to consider additional statistical metrics such as, for example, median for node constraints computations and see how the communication burden will be affected. Another direction would be to use a statistical approach in order to decide whether or not to disregard some local violations. By doing so, it may reduce the communication load without drastic degrading monitoring performance.

A possible application of classical clustering algorithms is an additional research direction that may lead to scalable clustering procedures. While the experimental results demonstrate that communication savings may depend on the choice of a norm or a distance function many of the proposed clustering algorithms can be applied with Bregman

divergences [2006].

Furthermore, another approach of considerable interest would be to search for the "largest" possible convex region in $\mathbf{Z}_+(f)$ that contains the convex hull of the vectors described in [2007]. So as long as the new vectors $\mathbf{v}_i(t)$ belong in this convex region no communication between the nodes will be required. We call this region "largest" because it provides more "freedom" to the new vectors $\mathbf{v}_i(t)$ while reducing the communication between the nodes. Finally based on this thesis we can conclude that the amount of communication required depends on the "trade off" parameter $0 < \alpha < 1$ selected at the beginning of the monitoring process. Dependence of the number of broadcasts on a is not understood at this point and should be further investigated. Figure 5.0.1 shows the number of broadcasts for 18 values $\alpha = 0.05, 0.10, \cdots, 0.95$. The smallest number of broadcasts corresponds to $\alpha = 0.30$ (as reported in Table 4.2.8). Next we run monitoring for 98 values of $\alpha = 0.01, 0.02, \cdots, 0.99$ (see Figure 5.0.2). This time the smallest number of broadcasts corresponds to $\alpha = 0.16$. Zooming in does not indicate any particularly useful property of the function.

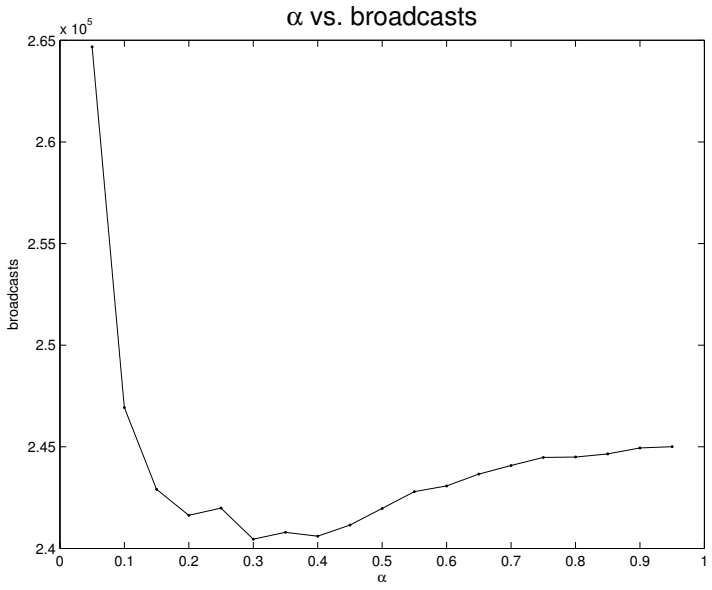


Fig. 5.0.1: $l_\infty, \alpha = 0.05, 0.10, \cdots, 0.95$ vs broadcasts, for feature "ipo"

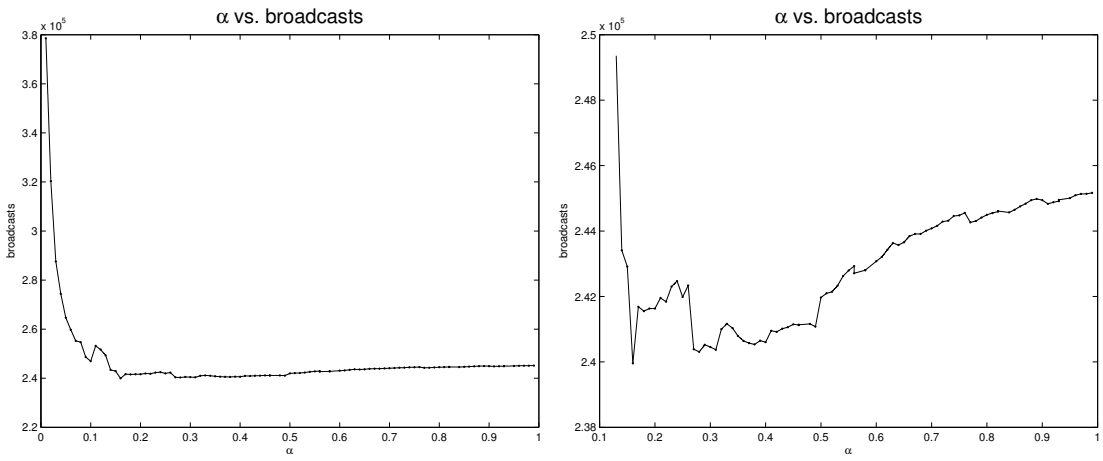


Fig. 5.0.2: feature "ipo", l_∞ norm, $\alpha = 0.01, 0.02, \cdots, 0.99$ vs broadcasts, (left) and zoom in for $\alpha = 0.14, 0.15, \cdots, 0.99$ (right)

Thus the approach of monitoring distributed data streams through node clustering, if successful, in many cases reduces the communication required to message exchanges within a cluster only, yielding overall communication reduction. While the results obtained show improvement over previously reported ones that do not use clustering [2012] it is of interest to introduce an update of α based on the monitoring history each time nodes are re–clustered.

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.

Appendices

Appendix A: First and Second Moments

In what follows we consider the auxiliary problem: "Let $\mathbf{X}$ be a vector set of size m with mean μ and variance $\gamma > 0$. How far away from μ a vector $\mathbf{x} \in \mathbf{X}$ can get?" The answer to this question is provided below. To simplify the exposition we first assume $\mu = 0$.

Let $\mathcal{X}(m, \gamma)$ be a family of sets $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbf{R}^d$ with $\mu(\mathbf{X}) = 0$, and $\sigma^2(\mathbf{X}) = \sum_{i=1}^m (\mathbf{x}_i - \mu(\mathbf{X}))^T (\mathbf{x}_i - \mu(\mathbf{X})) = \gamma \geq 0$. In this section $||\mathbf{x}||$ stands for $||\mathbf{x}||_2$. For each $\mathbf{X} \in \mathcal{X}(m, \gamma)$ define $r(\mathbf{X})$ and $R(\gamma)$ as follows:

$$r(\mathbf{X}) = \max_{\mathbf{x} \in \mathbf{X}} \|\mathbf{x}\|, \text{ and } R(\gamma) = \sup_{\mathbf{X} \in \mathcal{X}(m,\gamma)} r(\mathbf{X}).$$

In what follows we describe sets $\overline{\mathbf{X}}_\gamma \in \mathcal{X}(m, \gamma)$ that maximize r, and the function $R(\gamma)$. This implies that $r\left(\overline{\mathbf{X}}_\gamma\right) \geq r\left(\mathbf{X}\right)$, $\forall \mathbf{X} \in \mathcal{X}(m, \gamma)$ and $r\left(\overline{\mathbf{X}}_\gamma\right) = R(\gamma)$.

Lemma .1. *The function $R(\gamma)$ is a homogeneous function of degree $\frac{1}{2}$. For each positive scalar c one has $R(c\gamma) = c^{\frac{1}{2}} R(\gamma)$.*

Proof. Note that for positive scalars t and s one has

$$tR(\gamma) = tr\left(\overline{\mathbf{X}}_\gamma\right) = r\left(t\overline{\mathbf{X}}_\gamma\right) \leq r\left(\overline{\mathbf{X}}_{\gamma t^2}\right) = R(\gamma t^2),$$

and

$$sR(\gamma t^2) = sr(\overline{\mathbf{X}}_{\gamma t^2}) = r(s\overline{\mathbf{X}}_{\gamma t^2}) \leq r(\overline{\mathbf{X}}_{\gamma t^2 s^2}) = R(\gamma t^2 s^2).$$

In particular when $ts = 1$ one has

$$R(\gamma) \leq t^{-1} R(\gamma t^2), \text{ and } R(\gamma t^2) \leq s^{-1} R(\gamma t^2 s^2) = s^{-1} R(\gamma).$$

This shows that for positive t one has $tR(\gamma) = R(\gamma t^2)$ and completes the proof. □

Lemma .2. *Let* $\mathbf{u} \in \overline{\mathbf{X}}_\gamma$ *be such that* $\|\mathbf{u}\| = r(\overline{\mathbf{X}}_\gamma) = R(\gamma)$. *For each* $\mathbf{x} \in \overline{\mathbf{X}}_\gamma$ *there is a scalar* c *such that* $\mathbf{x} = c\mathbf{u}$.

Proof. We assume now that the claim is false. Without any loss of generality we assume that $\|\mathbf{u}\| = 1$. Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ be all nonzero vectors in $\overline{\mathbf{X}}_\gamma$ so that $\mathbf{u}(\mathbf{u}^T\mathbf{x}_i) \neq \mathbf{x}_i$, $i = 1, \ldots, k$. Since $\mu(\overline{\mathbf{X}}_\gamma) = 0$ and the vectors $\mathbf{x}_i - \mathbf{u}(\mathbf{u}^T\mathbf{x}_i) \neq 0$ then

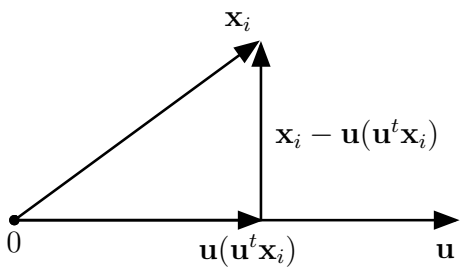


Fig. .0.3

$\sum_{i=1}^{k} \left[\mathbf{x}_i - \mathbf{u}(\mathbf{u}^T\mathbf{x}_i)\right] = 0$ and $\sum_{i=1}^{m} \mathbf{u}(\mathbf{u}^T\mathbf{x}_i) = 0$ (see Fig. .0.3). Consider next the vector set $\mathbf{X}' = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_m\}$ where

$$\mathbf{x}'_i = \frac{1}{2}[\mathbf{x}_i - \mathbf{u}(\mathbf{u}^T\mathbf{x}_i)] + \mathbf{u}(\mathbf{u}^T\mathbf{x}_i),\ i = 1, \ldots, k, \quad \text{and } \mathbf{x}'_i = \mathbf{x}_i,\ i = k+1, \ldots, m$$

(see Fig. .0.4).

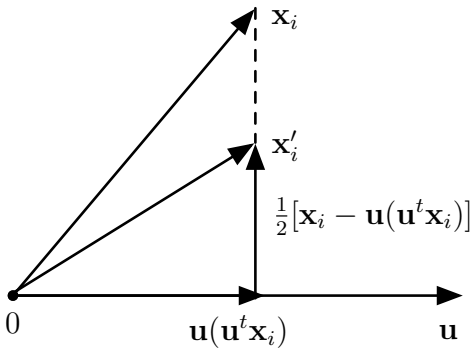


Fig. .0.4

We note that $\mu(\mathbf{X}') = 0$, and $\sigma^2(\mathbf{X}') = \gamma' < \gamma$. Due to Lemma .1 one has

$$1 = ||\mathbf{u}|| = r(\mathbf{X}') \leq R(\gamma') < R(\gamma) = ||\mathbf{u}|| = 1.$$

This contradiction completes the proof. □

Thus we proved that there is no vector $\mathbf{x}_i$ that sticks out of the line defined by the vector $\mathbf{u}$.

Lemma .3. *Let* $\mathbf{u} \in \overline{\mathbf{X}}_\gamma$ *be such that* $\|\mathbf{u}\| = r(\overline{\mathbf{X}}_\gamma) = R(\gamma)$. *For each* $\mathbf{x} \in \overline{\mathbf{X}}_\gamma$, $\mathbf{x} \neq \mathbf{u}$ *there is a scalar* $c \leq 0$ *such that* $\mathbf{x} = c\mathbf{u}$.

Proof. First note that there is at least one $\mathbf{x} \in \overline{\mathbf{X}}_\gamma$ such that $\mathbf{x} = c\mathbf{u}$ with $c < 0$. We denote this c by c_-. Assume that the statement of the lemma is false. Then there is $0 < c_+ \leq 1$ such that $c_+\mathbf{u} \in \overline{\mathbf{X}}_\gamma$ (see Fig. .0.5).

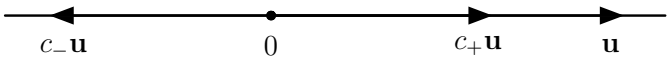


Fig. .0.5

Let $\epsilon > 0$ be so small that $c_+ - \epsilon > 0$, and $c_- + \epsilon < 0$ (see Fig. .0.6). Define $\mathbf{X}'$ by

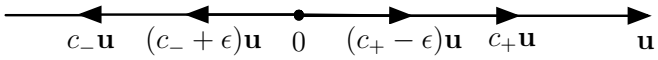


Fig. .0.6

substituting the vectors $c_+\mathbf{u}$ and $c_-\mathbf{u}$ by $(c_+ - \epsilon)\mathbf{u}$ and $(c_- + \epsilon)\mathbf{u}$ correspondingly, and keeping the other $m - 2$ vectors unchanged.

We note that $\mu(\mathbf{X}') = 0$, and $\sigma^2(\mathbf{X}') = \gamma' < \gamma$. Due to Lemma .1 one has

$$\|\mathbf{u}\| = r(\mathbf{X}') \leq R(\gamma') < R(\gamma) = \|\mathbf{u}\|.$$

This contradiction completes the proof. □

Thus all $\mathbf{x} \in \overline{\mathbf{X}}_\gamma$, $\mathbf{x} \neq \mathbf{u}$ are of the form $\mathbf{x} = c\mathbf{u}$ where $c \leq 0$. Next we prove that all these vectors $\mathbf{x}$ have the same length.

Lemma .4. *Let* $\mathbf{u} \in \overline{\mathbf{X}}_\gamma$ *be such that* $\|\mathbf{u}\| = r(\overline{\mathbf{X}}_\gamma) = R(\gamma)$. *If* $\mathbf{x} \in \overline{\mathbf{X}}_\gamma$ *and* $\mathbf{x} \neq \mathbf{u}$, *then* $\mathbf{x} = -\frac{1}{m-1}\mathbf{u}$.

Proof. Assume the opposite, i.e., there are $\mathbf{x}_1 = c_1\mathbf{u}$, and $\mathbf{x}_2 = c_2\mathbf{u}$ such that $c_1 < c_2 \leq 0$. Let $\mathbf{X}'$ be a vector set obtained from $\overline{\mathbf{X}}_\gamma$ by substituting $\mathbf{x}_1 = c_1\mathbf{u}$ by $\mathbf{x}'_1 = (c_1 + \epsilon)\mathbf{u}$, $\mathbf{x}_2 = c_2\mathbf{u}$ by $\mathbf{x}'_2 = (c_2 - \epsilon)\mathbf{u}$, and keeping the other vectors unchanged (see Fig. .0.7).

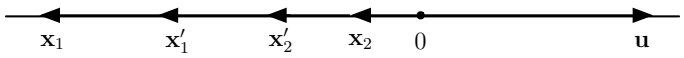


Fig. .0.7

Note that $\mu(\mathbf{X}') = 0$, and

$$\sigma^2(\overline{\mathbf{X}}_\gamma) - \sigma^2(\mathbf{X}') = 2\epsilon(c_2 - c_1 - \epsilon).$$

We note that for a small positive ϵ one has $\sigma^2(\overline{\mathbf{X}}_\gamma) > \sigma^2(\mathbf{X}')$. Due to Lemma .1 one has

$$\|\mathbf{u}\| = r(\mathbf{X}') \leq R(\gamma') < R(\gamma) = \|\mathbf{u}\|.$$

This contradiction completes the proof. □

The next statement summarizes the above results.

Theorem .5. *If* $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbf{R}^d$ *with* $\mu(\mathbf{X}) = \mu$, *and* $\sigma^2(\mathbf{X}) = \gamma \geq 0$, *then*

$$\|\mathbf{x}_i - \mu\|^2 \leq \frac{m-1}{m}\gamma.$$

Further, $\|\mathbf{x}_m - \mu\|^2 = \frac{m-1}{m}\gamma$ *if and only if*

$$\mathbf{x}_1 = \cdots = \mathbf{x}_{m-1} = \mu - \frac{1}{m-1}[\mathbf{x}_m - \mu].$$

Next we prove Theorem .5 for scalars and then for vectors. In both proofs we don't use any of the previous Lemmas but we use the rearrangement inequality.

Definition .6. For every choice of real numbers $\mathbf{x}_1 \leq \cdots \leq \mathbf{x}_n$ and $\mathbf{y}_1 \leq \cdots \leq \mathbf{y}_n$ and every permutation $\mathbf{x}_{\sigma(1)}, \cdots, \mathbf{x}_{\sigma(n)}$ of $\mathbf{x}_1, \cdots, \mathbf{x}_n$,

$$\mathbf{x}_n\mathbf{y}_1 + \cdots + \mathbf{x}_1\mathbf{y}_n \leq \mathbf{x}_{\sigma(1)}\mathbf{y}_1 + \cdots + \mathbf{x}_{\sigma(n)}\mathbf{y}_n \leq \mathbf{x}_1\mathbf{y}_1 + \cdots + \mathbf{x}_n\mathbf{y}_n.$$

If the numbers are different, meaning that $\mathbf{x}_1 < \cdots < \mathbf{x}_n$ and $\mathbf{y}_1 < \cdots < \mathbf{y}_n$, then the lower bound is attained only for the permutation which reverses the order, i.e. $\sigma(i) = n - i + 1$ for all $i = 1, \cdots, n$, and the upper bound is attained only for the identity, i.e. $\sigma(i) = i$ for all $i = 1, \cdots, n$.

Proof. (of Theorem .5)

Without loss of generality, $\mu = 0$ and $\mathbf{x}_1 \leq \mathbf{x}_2 \leq \cdots \leq \mathbf{x}_m$, where $\mathbf{x}_i$ are constants. Also assume that $|\mathbf{x}_1| \leq |\mathbf{x}_m|$. (The proof can be done similarly when $|\mathbf{x}_1| \geq |\mathbf{x}_m|$.) Then it is enough to show that

$$\mathbf{x}_m{}^2 \leq \tfrac{m-1}{m} \textstyle\sum_{i=1}^{m} \mathbf{x}_i{}^2 \text{ since } \gamma = \sum_{i=1}^{m} \mathbf{x}_i{}^2.$$

Starting with the left hand side of the inequality we have that

$$\mathbf{x}_m{}^2 = (-\mathbf{x}_1 - \mathbf{x}_2 - \cdots - \mathbf{x}_{m-1})^2 = \sum_{i=1}^{m-1} \mathbf{x}_i^2 + 2 \sum_{1 \leq i < j \leq m-1} \mathbf{x}_i\mathbf{x}_j \tag{.0.1}$$

provided $\mu = 0$. Next by using the rearrangement inequality we prove that

$$2 \sum_{1 \leq i < j \leq m-1} \mathbf{x}_i\mathbf{x}_j \leq (m-2) \sum_{i=1}^{m-1} \mathbf{x}_i{}^2. \tag{.0.2}$$

By Induction

- $m = 3 \Rightarrow 2\mathbf{x}_1\mathbf{x}_2 \leq \mathbf{x}_1{}^2 + \mathbf{x}_2{}^2$, which is true by rearrangement inequality.
- $m = 4$ then we need to show that $2(\mathbf{x}_1\mathbf{x}_2 + \mathbf{x}_1\mathbf{x}_3 + \mathbf{x}_2\mathbf{x}_3) \leq 2(\mathbf{x}_1{}^2 + \mathbf{x}_2{}^2 + \mathbf{x}_3{}^2)$. By starting with the left hand side we can see that there are 6 terms in total. Notice that

each term $\mathbf{x}_i$ appears 4 times for all $i = 1, \cdots, m-1$. By rearrangement inequality we have that each term appears twice. So we have $m-1$ terms that each appears twice in $m-2$ expressions, i.e, $4 = 2(m-2)$. Thus,

$$2 \sum_{1 \leq i < j \leq m-1} \mathbf{x}_i \mathbf{x}_j \leq (m-2)(\mathbf{x}_1{}^2 + \mathbf{x}_2{}^2 + \mathbf{x}_3{}^2). \tag{.0.3}$$

- $\cdots$
- m terms then we need to show that $2\sum_{1 \leq i < j \leq m-1} \mathbf{x}_i \mathbf{x}_j \leq (m-2)\sum_{i=1}^{m-1} \mathbf{x}_i{}^2$. Starting again with the left hand side we can see that there are $(m-1)(m-2)$ terms in the summation. Similarly, by rearrangement inequality each of the $m-1$ terms appears twice in $m-2$ expressions.

By using now .0.2 and .0.1 it is easy to see that

$$\mathbf{x}_m{}^2 \leq \frac{m-1}{m} \sum_{i=1}^{m} \mathbf{x}_i{}^2. \tag{.0.4}$$

□

Lemma .7. *If* $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbf{R}^d$ *such that*

1. $\mathbf{x}_1 + \cdots + \mathbf{x}_m = 0$,
2. $\mathbf{x}_1^t \mathbf{x}_1 + \cdots + \mathbf{x}_m^t \mathbf{x}_m = \gamma \geq 0$,

then $\mathbf{x}_m^t \mathbf{x}_m \leq \frac{m-1}{m}\gamma$.

Proof.

$$\begin{aligned}(\mathbf{x}_1 + \cdots + \mathbf{x}_{m-1})^t(\mathbf{x}_1 + \cdots + \mathbf{x}_{m-1}) &= \sum_{i=1}^{m-1} \mathbf{x}_i^t \mathbf{x}_i + \sum_{i \neq j}^{m-1} \mathbf{x}_i^t \mathbf{x}_j = \sum_{i=1}^{m-1} \mathbf{x}_i^t \mathbf{x}_i + 2\sum_{j>i}^{m-1} \mathbf{x}_i^t \mathbf{x}_j \\ &\leq \sum_{i=1}^{m-1} \mathbf{x}_i^t \mathbf{x}_i + (m-2)\sum_{i=1}^{m-1} \mathbf{x}_i^t \mathbf{x}_i = (m-1)\sum_{i=1}^{m-1} \mathbf{x}_i^t \mathbf{x}_i\end{aligned}$$

(this inequality follows from the rearrangement inequality).

By adding $(m-1)\mathbf{x}_m^t\mathbf{x}_m$ to both sides of the inequality one gets

$$m\mathbf{x}_m^t\mathbf{x}_m \leq (m-1)\sum_{i=1}^{m}\mathbf{x}_i^t\mathbf{x}_i = (m-1)\gamma.$$

This completes the proof. □

Appendix B: Broadcast Count

Transmission of a double precision real number is defined as a message in Kogan [2012]. In this thesis, 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, the sum of its coordinates adds up to 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 thesis. 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 1 triggered by violation of a node constraint. We first assume that the violator node $\mathbf{n}$ is different from the root.

1. The root notifies all other nodes (except the the violator node $\mathbf{n}$) 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 monitoring with clustering (Algorithm 3). The monitoring procedure starts with each node $\mathbf{n}$ sending its 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

$$2(n-1) \tag{.0.5}$$

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 is violated 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 its 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 brings the number of broadcasts to $2n-3$. If the violating node is the root, then this number is $2n-2$. To simplify the computations we select the largest number $2n-2$.

At this step, and keeping in mind (.0.5), the total number of broadcasts needed to be exchanged is

$$2(n-1)+2n-2=4(n-1). \tag{.0.6}$$

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 its 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 (.0.6), the total number of broadcasts right after the first root mean update and first clustering is

$$5(n-1). \tag{.0.7}$$

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 violating node $\mathbf{n}$ reports its 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 $\mathbf{h_n}$, 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 $\mathbf{h_n}$ 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 its 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 the violating node is not the root, and $3(n-1)$ broadcasts if the violation is at the root. To compute the broadcasts we use

the larger number

$$3(n-1). \tag{.0.8}$$

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

   (a) the violator $\mathbf{n}$ reports its ID, $\Delta_{\mathbf{n}} = \mathbf{v}_{\mathbf{n}}(t) - \mathbf{v}_{\mathbf{n}}(t_k)$, 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 $\Delta_{\mathbf{n}}$ and constraints $\delta(\mathbf{n})$ to the coordinator (total of $(|\pi| - 2)$ broadcasts). The total comes to $2|\pi| - 3$, and this number is $2|\pi| - 2$ when the violating node is the coordinator.

   The total of broadcasts needed is:

$$2|\pi| - 2. \tag{.0.9}$$

3. A coordinator constraint is violated. First we assume the coordinator $\mathbf{c}$ is not 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 $\mathbf{v}_{\mathbf{n}}(t)$, weights W_n, and history vectors $\mathbf{h}_{\mathbf{n}}$ ($n - 2$ 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

$$3(n-1). \tag{.0.10}$$

Bibliography

Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. Clustering with Bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, 2005.

Maria Barouti, Daniel Keren, Jacob Kogan, and Yaakov Malinovsky. Monitoring distributed data streams through node clustering. In *Machine Learning and Data Mining in Pattern Recognition*, pages 149–162. Springer, 2014a.

Maria Barouti, Daniel Keren, Jacob Kogan, and Yaakov Malinovsky. Adaptive clustering for monitoring distributed data streams. *Proceedings of the Workshop on Exploratory Data Analysis, (held in conjunction with the 2014 SIAM International Conference on Data Mining), SIAM, Philadelphia*, pages 13–16, 2014b.

Maria Barouti, Daniel Keren, Jacob Kogan, and Yaakov Malinovsky. Clustering for monitoring distributed data streams. In *Partitional Clustering Algorithms*, pages 387–415. Springer, 2015.

Daniel Boley. Principal direction divisive partitioning. *Data mining and knowledge discovery*, 2(4):325–344, 1998.

Peter Brucker. On the complexity of clustering problems. In *Optimization and operations research*, pages 45–54. Springer, 1978.

M Emre Celebi, Hassan A Kingravi, and Patricio A Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210, 2013.

Inderjit Dhillon, Jacob Kogan, and Charles Nicholas. Feature selection and document clustering. In *Survey of Text Mining*, pages 73–100. Springer, 2004.

Inderjit S Dhillon, Yuqiang Guan, and Jacob Kogan. Iterative clustering of high dimensional text data augmented by local search. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 131–138. IEEE, 2002.

Edwin Diday. The dynamic clusters method in nonhierarchical clustering. *International Journal of Computer & Information Sciences*, 2(1):61–88, 1973.

Mark Dilman and Danny Raz. Efficient reactive monitoring. *Selected Areas in Communications, IEEE Journal on*, 20(4):668–676, 2002.

Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

Edward W Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.

Robert M Gray. *Entropy and information theory*. Springer Science & Business Media, 2011.

Daniel Keren, Izchak Sharfman, Assaf Schuster, and Avishay Livne. Shape sensitive geometric monitoring. *Knowledge and Data Engineering, IEEE Transactions on*, 24 (8):1520–1535, 2012.

Jacob Kogan. Feature selection over distributed data streams through convex optimization. In *SDM*, pages 475–484. SIAM, 2012.

Jacob Kogan and Yaakov 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, 2013.

Stuart P Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

Samuel Madden and Michael J Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 555–566. IEEE, 2002.

A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. Finding (recently) frequent items in distributed data streams. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 767–778, April 2005. doi: 10.1109/ICDE.2005.68.

Boris Mirkin. *Clustering: a data recovery approach*. CRC Press, 2012.

Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

P. Semenov. Private communication, 2014.

Izchak Sharfman, Assaf Schuster, and Daniel Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Transactions on Database Systems (TODS)*, 32(4):23, 2007.

H Spath. Cluster dissection and analysis: theory, fortran programs, examples. horwood, 1985.

Helmuth Spath. *Cluster analysis algorithms for data reduction and classification of objects*. Ellis Horwood, Ltd. Chichester, England, 1980.

H Steinhaus. Sur la divisi on, des corps materials en perties. *Bulletin de LAcademie Polonaise des sciences*, 1956.

Marc Teboulle, Pavel Berkhin, I Dhillon, Yuqiang Guan, and Jacob Kogan. Clustering with entropy-like k-means algorithms. In *Grouping Multidimensional Data*, pages 127–160. Springer, 2006.