Scaling Geometric Monitoring Over Distributed Streams

Alexandros D. Keros

June 23, 2016

Supervised by: Prof. V.Samoladas

Table of contents

Data Stream Systems

- **Data streams**: Continuous, high volume, size unbound, violative, probably distributed
- *Pull paradigm*
- Centralizing and/or polling $\rightarrow$ prohibitive in terms of communication overhead
- Examples: telecommunication, sensor networks

The Geometric Monitoring Method

- Threshold monitoring
- Nodes communicate when needed
  - Local constraints
  - Violation resolution (*false alarms*)
- Arbitrary function monitoring
- Tight accuracy bounds
- A promising framework for *distributed data stream monitoring*

Motivation

Problems:

- increasing node population
- data volume
- data dimensionality
- arbitrary functions
- **communication - accuracy tradeoff**

Need for:

- scalability warranties
- tight accuracy bounds
- incremental/real-time operation
- **Minimize communication while retaining accuracy bounds**

Motivation

Problems:

- increasing node population
- data volume
- data dimensionality
- arbitrary functions
- **communication - accuracy tradeoff**

Need for:

- scalability warranties
- tight accuracy bounds
- incremental/real-time operation
- **Minimize communication while retaining accuracy bounds**

Motivation

Problems:

- increasing node population
- data volume
- data dimensionality
- arbitrary functions
- **communication - accuracy tradeoff**

Need for:

- scalability warranties
- tight accuracy bounds
- incremental/real-time operation
- **Minimize communication while retaining accuracy bounds**

Motivation

Problems:

- increasing node population
- data volume
- data dimensionality
- arbitrary functions
- **communication - accuracy tradeoff**

Need for:

- scalability warranties
- tight accuracy bounds
- incremental/real-time operation
- **Minimize communication while retaining accuracy bounds**

Contributions

Expand the *geometric monitoring method*:

- heuristic method for violation resolution
- distance-based hierarchical node clustering
- throughout method evaluation on synthetic and real-world datasets

Contributions

Expand the *geometric monitoring method*:

- heuristic method for violation resolution
- distance-based hierarchical node clustering
- throughout method evaluation on synthetic and real-world datasets

Contributions

Expand the *geometric monitoring method*:

- heuristic method for violation resolution
- distance-based hierarchical node clustering
- throughout method evaluation on synthetic and real-world datasets

Geometric Threshold Monitoring

- **Threshold monitoring**: arbitrary function $f(\cdot)$, threshold T

$$f(\cdot) < T \text{ or } f(\cdot) > T$$

- **Idea**: decompose into local constraints at the nodes

System Architecture

Decentralized Scenario

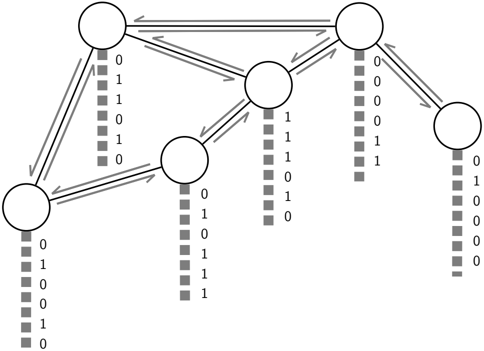


Figure: **Mesh-like network topology** example of the decentralized scenario. Dashed lines represent data streams and half arrows represent message exchanges.

System Architecture

Centralized Scenario

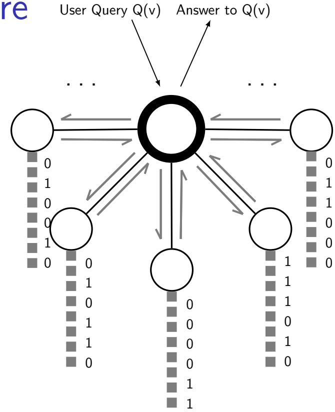


Figure: **Star-like network topology** example of the centralized scenario.The bold node represents the coordinator node. Dashed lines represent data streams and half arrows represent message exchanges.

Computational Model

Statistics vectors

- the *monitoring function* $f : \mathbb{R}^d \to \mathbb{R}$
- the *threshold* $T \in \mathbb{R}$
- the *monitoring node set* : $P = \{p_1, \ldots, p_n\}$
  with *weights* $w_1, \ldots, w_n$
- the *data streams* : $S = \{s_1, \ldots, s_n\}$
- the d-dimensional *local statistics vectors* : $\vec{v_1}(t), \ldots, \vec{v_n}(t)$
  represent each node's data stream at time t

Global statistics vector

$$\vec{v}(t) = \frac{\sum_{i=1}^{n} w_i \vec{v_i}(t)}{\sum_{i=1}^{n} w_i} \tag{1}$$

Computational Model

Estimate vector

Infrequent communication between nodes/nodes-coordinator:

Estimate vector

$$\vec{e}(t) = \frac{\sum_{i=1}^{n} w_i \vec{v_i}'}{\sum_{i=1}^{n} w_i} \tag{2}$$

- the last communicated *local statistics vector* of node p_i : $\vec{v_i}'$
- *Local statistics* divergence: $\Delta\vec{v_i}(t) = \vec{v_i}(t) - \vec{v_i}', i = 1, \ldots, n$

Decentralized drift vector

$$\vec{u_i}(t) = \vec{e}(t) + \Delta\vec{v_i}(t) \tag{3}$$

Centralized drift vector

$$\vec{u_i}(t) = \vec{e}(t) + \Delta\vec{v_i}(t) + \frac{\vec{\delta}_i}{w_i} \tag{4}$$

Computational Model

Balancing Process

Centralized scenario

Purpose: resolve possible false alarms

Balancing vector

$$\vec{b} = \frac{\sum_{p_i \in P'} w_i \vec{u_i}(t)}{\sum_{p_i \in P'} w_i} \quad (5)$$

- the *balancing set* P': a subset of nodes
- the *slack vector* at the nodes $\vec{\delta_i} = \vec{\delta_i}' + \Delta\vec{\delta_i}$, $\sum_{p_i \in P'} \Delta\vec{\delta_i} = \vec{0}$:

$$\Delta\vec{\delta_i} = w_i \vec{b} - w_i \vec{u_i}(t) \; \forall \; p_i \in P' \quad (6)$$

, readjusts the *drift vectors* (4).

Geometric Interpretation

Convexity Property

Convexity Property

$$\vec{v}(t) = \frac{\sum_{i=1}^{n} w_i \vec{u_i}(t)}{\sum_{i=1}^{n} w_i} \tag{7}$$

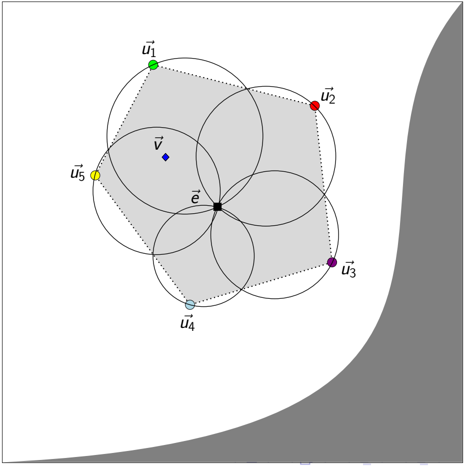
Theorem (Sharfman et al.)

Let $\vec{x}, \vec{y_1}, \ldots, \vec{y_n} \in \mathbb{R}^d$ be a set of vectors in $\mathbb{R}^d$. Let $Conv(\vec{x}, \vec{y_1}, \ldots, \vec{y_n})$ be the convex hull of $\vec{x}, \vec{y_1}, \ldots, \vec{y_n}$. Let $B(\vec{x}, \vec{y_i})$ be a ball centered at $\frac{\vec{x}+\vec{y_i}}{2}$ and with radius of $\|\frac{\vec{x}+\vec{y_i}}{2}\|_2$ i.e., $B(\vec{x}, \vec{y_i}) = \{\vec{z} \mid \|\vec{z} - \frac{\vec{x}+\vec{y_i}}{2}\|_2 \leq \|\frac{\vec{x}+\vec{y_i}}{2}\|_2\}$, then $Conv(vecx, \vec{y_1}, \ldots, \vec{y_n}) \subset B(\vec{x}, \vec{y_i})$.

Geometric Interpretation

Convexity Property & Local Constraints

Figure: Example of a convex hull (light gray) defined by the drift vectors $\vec{u_i}, i = 1, 2, 3, 4, 5$. The hull is bounded by the spheres created from the estimate vector $\vec{e}$ and the drift vectors $\vec{u_i}, i = 1, 2, 3, 4, 5$. The global statistics vector $\vec{v}$ is guaranteed to be contained in the convex hull of the drift vectors.



Protocol

Decentralized Algorithm

Algorithm 1: Decentralized algorithm

begin
  foreach *node* p_i **do**   `/* Node initialization */`
    Broadcast $\vec{v_i}(0)$;
    $\vec{v_i}' = \vec{v_i}(0)$;
    Wait messages from all other nodes;
    if *messages from all vectors received* **then**
      Compute *estimate vector* $\vec{e}(t)$;
    end
  end
  foreach *node* p_i **do**   `/* Main monitoring task */`
    foreach *new* s_i *stream update* $\vec{v_i}(t)$ **do**
      Recalculate *drift vector* $\vec{u_i}(t)$;
      if $B(\vec{e}, \vec{u_i}(t))$ *is not monochromatic* **then**
        Broadcast message $< i, \vec{v_i}(t) >$;
        Set $\vec{v_i}' = \vec{v_i}(t)$;
      end
      if *new message* $< j, \vec{v_j}(t) >$ *received* **then**
        Set $\vec{v_j}' = \vec{v_j}(t)$;
        Recalculate *estimate vector* $\vec{e}(t)$;
        if $B(\vec{e}, \vec{u_i}(t))$ *is not monochromatic* **then**
          Broadcast message $< i, \vec{v_i}(t) >$;
          Set $\vec{v_i}' = \vec{v_i}(t)$;
        end
      end
    end
  end
end

Protocol

Centralized Algorithm

Algorithm 2: Centralized algorithm's coordinator node operation

begin
  Wait for $< INIT, \cdot >$ messages from all monitoring nodes;
  /* Initialization */
  Compute *estimate vector* $\vec{e}(0)$;
  if *new* $< REP, \vec{v_i}(t), \vec{u_i}(t) >$ *message received* **then**
  /* Monitoring operation */
    $P' = P' \cup \{< i, \vec{v_i}(t), \vec{u_i}(t) >\}$;
    `Balance`(P');
  end
end
Function `Balance`(P') /* Balancing Process */
  Compute *balancing vector* $\vec{b}$;
  if $B(\vec{e}, \vec{b})$ *is not monochromatic* **then**
    if $P - P' \neq \emptyset$ **then**
      Send $< REQ >$ message to random node in $P - P'$ set;
    else
      Compute *estimate vector* $\vec{e}(t)$;
      Send $< NEW\text{-}EST, \vec{e}(t) >$ message to all nodes;
      return;
    end
  else
    foreach $p_i \in P'$ **do**
      Compute *slack adjustment vector* $\Delta\vec{\delta_i}$;
      Send $< ADJ\text{-}SLK, \Delta\vec{\delta_i} >$ message to node p_i;
      return;
    end
  end
end

Algorithm 3: Centralized algorithm's monitoring node operation

begin
  foreach *node* p_i **do** /* Node initialization */
    Send $< INIT, \vec{v_i}(0) >$ message to coordinator;
    $\vec{v_i}' = \vec{v_i}(0)$;
    $\vec{\delta_i} = \vec{0}$;
    Wait message from coordinator;
    if $< NEW\text{-}EST, \vec{e} >$ *message received* **then**
      Set $\vec{e}(t) = \vec{e}$;
    end
  end
  foreach *node* p_i **do** /* Main monitoring task */
    foreach *new* s_i *stream update* $\vec{v_i}(t)$ **do**
      Recalculate *drift vector* $\vec{u_i}(t)$;
      if $B(\vec{e}, \vec{u_i}(t))$ *is not monochromatic* **then**
        Send $< REP, \vec{v_i}(t), \vec{u_i}(t) >$ message to coordinator;
        Wait for $< NEW\text{-}EST, \cdot >$ or $< ADJ\text{-}SLK, \cdot >$ message from coordinator;
      end
      if *new message* $< REQ >$ *received* **then**
        Send $< REP, \vec{v_i}(t), \vec{u_i}(t) >$ message to coordinator;
        Wait for $< NEW\text{-}EST, \cdot >$ or $< ADJ\text{-}SLK, \cdot >$ message from coordinator;
      end
      if *new* $< NEW\text{-}EST, \vec{e} >$ *message received* **then**
        Set $\vec{e}(t) = \vec{e}$;
        $\vec{v_i}' = \vec{v_i}(t)$;
        $\vec{\delta_i} = \vec{0}$;
      end
      if *new* $< ADJ\text{-}SLK, \Delta\vec{\delta_i} >$ *message received* **then**
        Recompute *delta vector* $\vec{\delta_i}$;
      end
    end
  end
end

Multi-objective Optimization

- **Multiple**, possibly **conflicting** objectives to be *simultaneously* optimized
- *Pareto optimality*(*non-dominated solutions*): optimal solutions where none of the objective functions can be optimized without the simultaneous degradation of other objective functions' values.
- Let vector of m objectives $F(x) = [F_1(x), F_2(x), \ldots, F_m(x)]$:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} F(x) \\ &\text{s.t. } l \leq x \leq u \\ &\qquad G_i = 0, i = 1, \ldots, k_e \\ &\qquad G_j \leq 0, j = k_e + 1, \ldots, k \end{aligned}$$

- Finding *Pareto optimal* solutions is generally **NP-hard**.

Non-linear Constraint Optimization

Primal Descent

Algorithm 4: Generic primal descent

```
begin
    Choose initial point x_0 ∈ X and set t = 0 ;          /* Initialization */
    while maximum iteration limit OR convergence do        /* Search */
        t = t + 1;
        Determine search direction d_t;
        Determine step length s_t, so that f(x_t + s_t d_t) < f(x_t);
        Update;
    end
end
```

Feasible Directions

Usable feasible direction d_t:

- a small disposition towards direction d_t does not violate any constraint i.e.,

$$d_t^T \nabla G(x_t) \leq 0$$

- a move towards d_t reduces the objective functions value i.e.,

$$d_t^T \nabla F(x_t) < 0$$

.

SQP

- The *Lagrangian function*: $\mathcal{L}(x,\lambda) = F(x) + \sum_{i=1}^{k} \lambda_i G_i(x)$
- *Quadratic programming subproblems*:

$$\min_{d \in \mathbb{R}^n} \frac{1}{2} d^T H_t d + \nabla F(x_t)^T d$$
$$\nabla G_i(x_t)^T d + G_i(x_t) = 0, i = 1, \ldots, k_e$$
$$\nabla G_i(x_t)^T d + G_i(x_t) \leq 0, i = k_e + 1, \ldots k$$

,where:

H_t:Hessian of the Lagrangian function at iteration t

d:search direction

The Savitzky-Golay Filter

- Low-pass smoothing filter
- *Moving window averaging* paradigm: $g_i = \sum_{n=-n_L}^{n_R} c_n f_{i+n}$
- *Least-squares fit* of polynomial $y_i(x)$ over window $n_L + n_R + 1$:

$$\sum_{j=i-n_L}^{i+n_R} (y_i(x_j) - f_j)^2 = \min$$

,where:

$$y_i(x) = a_0 + a_1 \frac{x - x_i}{\Delta x} + a_2 (\frac{x - x_i}{\Delta x})^2 + \cdots + a_M (\frac{x - x_i}{\Delta x})^M$$

- Set g_i to the value of the fitted point x_i

Maximum Weight Matching

Let $G = (V, E)$ a graph:

- *maximum weight matching* $M \subseteq E$: a subset of edges where
  - no two edges share a common vertex
  - largest possible number of edges
  - maximizes the sum of weights

Maximum Weight Matching

The Primal-Dual Method

Primal-Dual Method

$$\text{Constraints in Primal} \iff \text{Variables in Dual}$$
$$\text{Constraints in Dual} \iff \text{Variables in Primal}$$

The primal:

$$\begin{aligned} \max \quad & \sum_{(u,v)\in E} x_{u,v} w_{u,v} & \\ \text{s.t.} \quad & x_{u,v} \geq 0 & ,(u,v) \in E \\ & \sum_{u\in e: e\in E} x_e \leq 1 & ,u \in V \end{aligned}$$

The dual:

$$\begin{aligned} \min \quad & \sum_{u\in V} y_u & \\ \text{s.t.} \quad & y_u \geq 0 & ,u \in V \\ & y_u + y_v \geq w_{u,v} & ,(u,v) \in E \end{aligned}$$

Related Work

- *Safe Zones*: optimal local constraints fitted to nodes' data distributions
- Ellipsoidal bounding regions, decouplement of estimate vector from bounding ball construction
- Simple shapes as local constraints, hierarchical clustering of nodes for participation to the balancing operation
- Prediction models based on velocity and acceleration

Problem Formulation

Reduce the communication burden of the *Geometric Monitoring* method by:

- Optimally position *drift vectors* during the *balancing process*
- Appropriate node selection for inclusion in the *balancing set*

,in order to **increase scalability** in terms of:

- node popullation
- stream dimensionality.

The Geometric Monitoring Framework

Assumptions

- Coordinator-based scenario
- Instantaneous, loss-less, reliable communication
- Iterative operation based on time-steps
- System pause during violation resolution
- Coordinator node does not monitor a stream

Algorithm 5: Iterative network operation

Data: *monitoringNodes*: a list of Monitoring nodes, *coordinator*: the Coordinator node

```
begin
    initialization;
    repeat
        foreach node ∈ monitoringNodes do
            node.DataVectorUpdate();
            node.ComputeDriftVector();
        end
        foreach node ∈ monitoringNodes do
            node.CheckForViolation();
            if localViolation then
                node.Report();
                coordinator.Balance();
            end
        end
    until globalViolation;
end
```

The Distance-based Hierarchical Clustering

The Idea

Node selection for *violation resolution*:

- Elevate *randomness* of the initial *geometric monitoring* method
- Hierarchical node clustering scheme
- Decouple matching from the data distribution at the nodes
- Accurately follow *global statistics vector*

The Distance-based Hierarchical Clustering

The Weight Function

Weight function

$$w_{i,j} = \sum_{t=t_0}^{t_{end}} [(f(\vec{v}_{global}(t)) - f(\frac{\vec{v}_i(t) + \vec{v}_j(t)}{2})) + (|\vec{v}_i(t) - \vec{v}_j(t)|)]$$

The Distance-based Hierarchical Clustering

The Algorithm

Algorithm 6: Recursively create Monitoring node pairs and hierarchy

Function `DistancePairer`($nodes$,i)

  Data: $nodes = [(n_1, [\vec{v_1}(t_0), ..., \vec{v_1}(t_{end})]), ..., (n_k, [\vec{v_k}(t_0), ..., \vec{v_k}(t_{end})])]$: list of nodes with their respective data vectors, i: pair type, initial=1

  Result: $nodeHierarchy$: dictionary of *Type-k* pairs

  if $length(nodes) = 1$ **then** `// recursion stopping condition`

    return $nodeHierarchy$;

  end

  $g = CreateCompleteGraph(nodes)$; `// complete graph with` *nodes* `as vertices`

  foreach $(n_i, n_j) \in g.Edges()$ **do** `// assign weights to edges`

    $w_{i,j} = \sum_{t=t_0}^{t_{end}} [(f(\vec{v}_{global}(t)) - f(\frac{\vec{v_i}(t)+\vec{v_j}(t)}{2})) + (|\vec{v_i}(t) - \vec{v_j}(t)|)]$;

    $g.edge(n_i, n_j).weight = w_{i,j}$;

  end

  $nodeHierarchy(\text{Type-i}) = g.maximalWeightMatching()$; `// node pairs of` *Type-i*

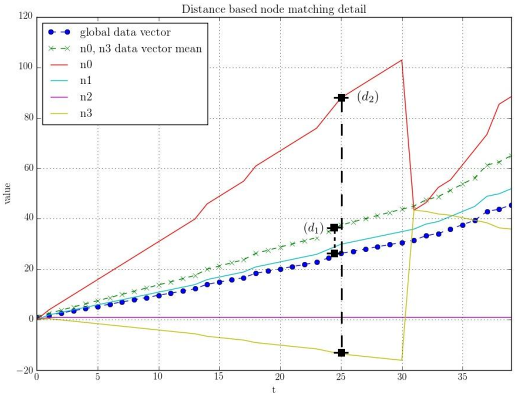
  `DistancePairer`($nodeHierarchy(Type\text{-}i), i * 2$);

end

The Distance-based Hierarchical Clustering

Example

Figure: Distance based node matching operating on 4 nodes ($\{n_0, n_1, n_2, n_3\}$). Distance d_1 : the distance of the data vector mean of the paired nodes n_0 and n_3 from the global mean (*global data vector*), distance d_2 : denotes the in-between distance of data vectors $\vec{v_0}(t)$ and $\vec{v_3}(t)$ of the node pair. Both distances are taking part in the edge weighting process.



The Heuristic Balancing

The Idea

The Heuristic Balancing

The Optimizing Function

The Heuristic Balancing

The Function Formulation

The Heuristic Balancing

The Algorithm

An Nested Optimization Problem

Velocity and Acceleration Estimation via SG Filtering

Implementation Challenges

Synthetic Data

Real-world Data

Notation

RAND, DIST, DISTR Comparison

GM, HM Comparison

GM, HDM Comparison

Synthetic Data Monitoring

GM, HDM Comparison

Air Pollution Monitoring

Summary & Concluding Remarks

Future Work

The end
Questions?

Appendix

Savitzky-Golay filter matrix notation

hey you