Distributed Monitoring of the R^2 Statistic for Linear Regression

Kanishka Bhaduri[1], Kamalika Das[2], Chris Giannella[3]

[1]MCT Inc@NASA Ames, [2]SGT Inc@NASA Ames, [3]MITRE Corp

TU Dortmund Seminar

July 14, 2011

Roadmap

Input (X)			Output ($\overrightarrow{y}$)
$x_{1,1}$	$x_{1,2}$	$\cdots$	y_1
$x_{2,1}$	$x_{2,2}$	$\cdots$	y_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Introduction

Input (X)			Output ($\overrightarrow{y}$)
$x_{1,1}$	$x_{1,2}$	$\cdots$	y_1
$x_{2,1}$	$x_{2,2}$	$\cdots$	y_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$

- Learn f: a function from $\mathbb{R}^d$ to $\mathbb{R}$
- For linear models, $\hat{y} = f(\overrightarrow{x}) = a_0 + x_{\cdot,1} * a_1 + x_{\cdot,2} * a_2 + x_{\cdot,3} * a_3 + \ldots$

Introduction

Input (X)			Output ($\overrightarrow{y}$)
$x_{1,1}$	$x_{1,2}$	$\cdots$	y_1
$x_{2,1}$	$x_{2,2}$	$\cdots$	y_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$

- Learn f: a function from $\mathbb{R}^d$ to $\mathbb{R}$
- For linear models, $\hat{y} = f(\overrightarrow{x}) = a_0 + x_{\cdot,1} * a_1 + x_{\cdot,2} * a_2 + x_{\cdot,3} * a_3 + \ldots$

$$\overrightarrow{a} = (X^T X)^{-1} X^T y$$

Goodness-of-fit: coefficient of determination R^2

R^2 statistic

$$R^2 = 1 - \frac{\sum_{j=1}^{M} (y_j - \hat{y}_j)^2}{\sum_{j=1}^{M} \left(y_j - \frac{\sum_{j=1}^{n} y_j}{M} \right)^2}$$

Goodness-of-fit: coefficient of determination R^2

R^2 statistic

$$R^2 = 1 - \frac{\sum_{j=1}^{M} (y_j - \hat{y}_j)^2}{\sum_{j=1}^{M} \left(y_j - \frac{\sum_{j=1}^{n} y_j}{M} \right)^2}$$

- For perfect models, $y_j = \hat{y}_j \Rightarrow R^2 = 1$
- For bad (average) models, $\hat{y}_j = \frac{\sum_{j=1}^{n} y_j}{M} \Rightarrow R^2 = 0$

What's new?

- The data is distributed across a large number of sites/nodes/machines
  - $X = X_1 \cup \cdots \cup X_n$
- X is time varying

Figure: Geographically distributed data centers (Image source: `http://verycloud.com/default.aspx`)

Motivation

- Real-time fault detection/health assessment in enterprise-scale data centers on the cloud
  - Microsoft Cloud computing infrastructure, Amazon EC2
- Carbon footprint monitoring for Smart grid connected infrastructures
- Useful for distributed systems with very little centralized control

Problem statement and solutions

- Given:
  - data $X_1, \ldots, X_n$ at each node
  - threshold $0 < \epsilon < 1$ (same at all nodes)
  - a reliable, ordered, communication infrastructure
  - precomputed $\overrightarrow{a}$ at an earlier timestamp
- Maintain:
  - regression model $\overrightarrow{a}$, such that $R^2 > \epsilon$

Problem statement and solutions

- Given:
  - data $X_1, \ldots, X_n$ at each node
  - threshold $0 < \epsilon < 1$ (same at all nodes)
  - a reliable, ordered, communication infrastructure
  - precomputed $\overrightarrow{a}$ at an earlier timestamp
- Maintain:
  - regression model $\overrightarrow{a}$, such that $R^2 > \epsilon$

Solutions

- Periodic algorithms
- Incremental algorithms
- Reactive event-based algorithms

Problem statement and solutions

- Given:
  - data $X_1, \ldots, X_n$ at each node
  - threshold $0 < \epsilon < 1$ (same at all nodes)
  - a reliable, ordered, communication infrastructure
  - precomputed $\overrightarrow{a}$ at an earlier timestamp
- Maintain:
  - regression model $\overrightarrow{a}$, such that $R^2 > \epsilon$

Solutions

- Periodic algorithms: wastes resources
- Incremental algorithms: most efficient, difficult to develop
- Reactive event-based algorithms: simple, yet efficient

Reformulating R^2 statistic

- Define a local vector $\overrightarrow{v^i}$ at node P_i based on y and $\hat{y}$
- Define a parabola $g : \overrightarrow{\beta} \in \mathbb{R}^2 \mapsto \beta_1 - (1-\epsilon)\beta_2^2$

Reformulating R^2 statistic

- Define a local vector $\overrightarrow{v^i}$ at node P_i based on y and $\hat{y}$
- Define a parabola $g : \overrightarrow{\beta} \in \mathbb{R}^2 \mapsto \beta_1 - (1-\epsilon)\beta_2^2$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \sum_{j=1}^{m(i)} (y_j^i - \hat{y}_j^i)^2}{\sum_{i=1}^{n} \sum_{j=1}^{m(i)} \left(y_j^i - \frac{\sum_{i=1}^{n} \sum_{j=1}^{m(i)} y_j^i}{M} \right)^2}$$

Reformulating R^2 statistic

- Define a local vector $\overrightarrow{v^i}$ at node P_i based on y and $\hat{y}$
- Define a parabola $g : \overrightarrow{\beta} \in \mathbb{R}^2 \mapsto \beta_1 - (1-\epsilon)\beta_2^2$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \sum_{j=1}^{m(i)} (y_j^i - \hat{y}_j^i)^2}{\sum_{i=1}^{n} \sum_{j=1}^{m(i)} \left(y_j^i - \frac{\sum_{i=1}^{n} \sum_{j=1}^{m(i)} y_j^i}{M} \right)^2}$$

$$\begin{aligned} R^2 > \epsilon \quad & \Leftrightarrow \quad g\left(\sum_{i=1}^{n} \left(\frac{m(i)}{M} \right) \overrightarrow{v^i} \right) > 0 \\ & \Leftrightarrow \quad g\left(\overrightarrow{v^G} \right) > 0 \end{aligned}$$

Reformulating R^2 statistic

- Define a local vector $\overrightarrow{v^i}$ at node P_i based on y and $\hat{y}$
- Define a parabola $g : \overrightarrow{\beta} \in \mathbb{R}^2 \mapsto \beta_1 - (1-\epsilon)\beta_2^2$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \sum_{j=1}^{m(i)} (y_j^i - \hat{y}_j^i)^2}{\sum_{i=1}^{n} \sum_{j=1}^{m(i)} \left(y_j^i - \frac{\sum_{i=1}^{n} \sum_{j=1}^{m(i)} y_j^i}{M} \right)^2}$$

$$R^2 > \epsilon \quad \Leftrightarrow \quad g\left(\sum_{i=1}^{n} \left(\frac{m(i)}{M} \right) \overrightarrow{v^i} \right) > 0$$

$$\Leftrightarrow \quad g\left(\overrightarrow{v^G} \right) > 0$$

Still inefficient to compute $\overrightarrow{v^G}$ at each timestamp...

Geometric interpretation

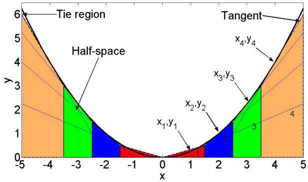


Figure: g and half-spaces

- $\overrightarrow{v^i}$ inside parabola for all $P_i \Rightarrow \overrightarrow{v^G}$ is inside too
- Not true for outside parabola
- Use tangent lines to define half spaces outside parabola

Geometric interpretation

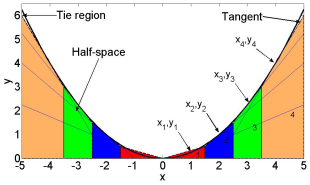


Figure: g and half-spaces

- $\overrightarrow{v^i}$ inside parabola for all $P_i \Rightarrow \overrightarrow{v^G}$ is inside too
- Not true for outside parabola
- Use tangent lines to define half spaces outside parabola

How to check for this global condition more efficiently?

Local statistics vectors

- Knowledge: $\overrightarrow{\mathcal{K}_i} = \overrightarrow{v^i} + \sum_{P_j \in N_i} \overrightarrow{S_{j,i}}$
- Agreement: $\overrightarrow{\mathcal{A}_{i,j}} = \overrightarrow{S_{i,j}} + \overrightarrow{S_{j,i}}$
- Withheld: $\overrightarrow{\mathcal{H}_{i,j}} = \overrightarrow{\mathcal{K}_i} - \overrightarrow{\mathcal{A}_{i,j}}$
- Message: $\overrightarrow{S_{i,j}} = \overrightarrow{\mathcal{K}_i} - \overrightarrow{S_{j,i}}$

Local statistics vectors

- Knowledge: $\overrightarrow{\mathcal{K}_i} = \overrightarrow{v^i} + \sum_{P_j \in N_i} \overrightarrow{S_{j,i}}$
- Agreement: $\overrightarrow{\mathcal{A}_{i,j}} = \overrightarrow{S_{i,j}} + \overrightarrow{S_{j,i}}$
- Withheld: $\overrightarrow{\mathcal{H}_{i,j}} = \overrightarrow{\mathcal{K}_i} - \overrightarrow{\mathcal{A}_{i,j}}$
- Message: $\overrightarrow{S_{i,j}} = \overrightarrow{\mathcal{K}_i} - \overrightarrow{S_{j,i}}$

We show: $g\left(\overrightarrow{\mathcal{K}_i}\right) > 0 \Rightarrow g\left(\overrightarrow{v^G}\right) > 0$....removing the sum inside parabola

Global condition

For each peer and each of its neighbors, if

- $\overrightarrow{\mathcal{K}_i} \in R$
- $\overrightarrow{\mathcal{A}_{i,j}} \in R$
- $\overrightarrow{\mathcal{H}_{i,j}} \in R$

where R is any convex region, then $\overrightarrow{v^G} \in R$

Global condition

For each peer and each of its neighbors, if

- $\overrightarrow{\mathcal{K}_i} \in R$
- $\overrightarrow{\mathcal{A}_{i,j}} \in R$
- $\overrightarrow{\mathcal{H}_{i,j}} \in R$

where R is any convex region, then $\overrightarrow{v^G} \in R$

Global condition $g\left(\overrightarrow{v^G}\right) \in R$ can be detected based solely on local conditions...

Convex regions

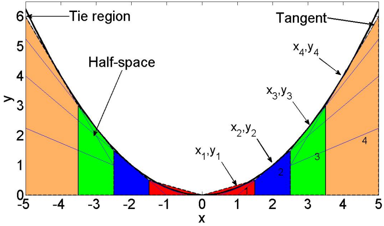


Figure: g and half-spaces

- Inside of parabola: $g(\overrightarrow{\beta}) > 0$, convex by definition
- Outside of parabola: define tangent lines to parabola $g(\overrightarrow{\beta}) = 0$, convex by construction

Local criterion

- Allows a node to terminate computation and communication whenever stopping condition is satisfied irrespective of other conditions
- Still guarantees eventual correctness
- Remarkably efficient in pruning messages
- Allows a node to sit idle until an event occurs:
  - send or receive message
  - change in local data X_i
  - change in node neighborhood

Monitoring algorithm:

1. Input: local dataset, error threshold ϵ
2. Goal: monitor $R^2 \gtrless \epsilon$
3. Initialization
   - $\overrightarrow{v_i}$
   - Compute sufficient statistics vectors
   - Define convex regions

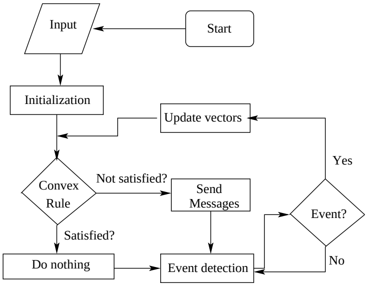


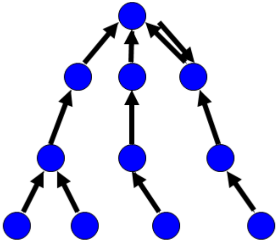
Figure: DReMo flowchart

Re-computing models: convergecast

- Monitoring algorithm raises an alarm on correct detection
- For closed-loop solution, rebuild model in-network
- Correctness of monitoring algorithm minimizes false dismissals and false alarms

$$X^T X = \sum_{i=1}^{n} X_i^T X_i \qquad X^T y = \sum_{i=1}^{n} X_i^T y_i$$

$$\vec{a} = \left(\sum_{i=1}^{n} X_i^T X_i\right)^{-1} \left(\sum_{i=1}^{n} X_i^T y_i\right)$$



Re-computing models: convergecast

- Monitoring algorithm raises an alarm on correct detection
- For closed-loop solution, rebuild model in-network
- Correctness of monitoring algorithm minimizes false dismissals and false alarms

$$X^T X = \sum_{i=1}^{n} X_i^T X_i \qquad X^T y = \sum_{i=1}^{n} X_i^T y_i$$

$$\vec{a} = \left(\sum_{i=1}^{n} X_i^T X_i \right)^{-1} \left(\sum_{i=1}^{n} X_i^T y_i \right)$$

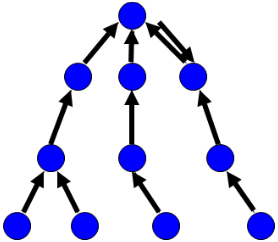


Figure: Convergecast

Experimental results: setup

- Each peer has a fixed length data buffer (a sliding window)

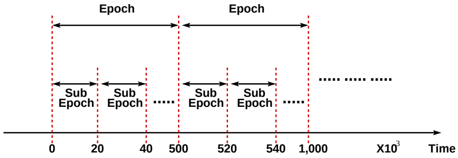


Figure: Timing diagram

Experimental results: no convergecast

- For every odd epochs, peers are supplied same model as data generator (high R^2 value)
- For every even epochs, peers are supplied a different model as data generator (low R^2 value)

Experimental results: no convergecast

- For every odd epochs, peers are supplied same model as data generator (high R^2 value)
- For every even epochs, peers are supplied a different model as data generator (low R^2 value)

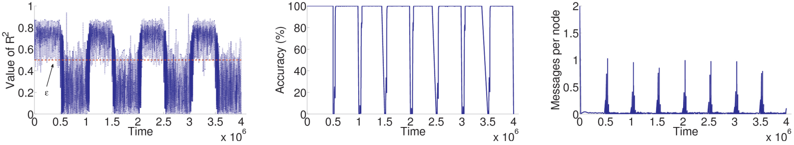


Figure: Dataset, accuracy and messages for *DReMo* algorithm in monitoring mode.

Experimental results: convergence and scalability

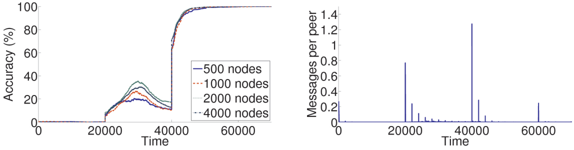


Figure: Convergence rate

Experimental results: convergence and scalability

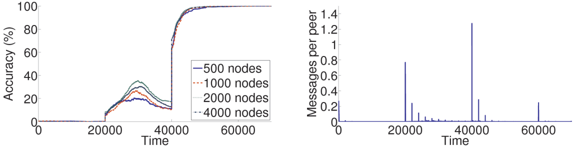


Figure: Convergence rate

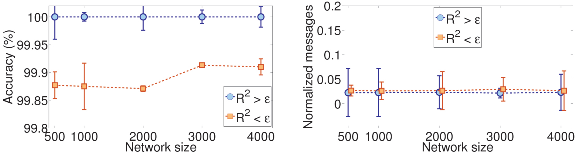


Figure: Scalability of *DReMo*

Experimental results: with convergecast

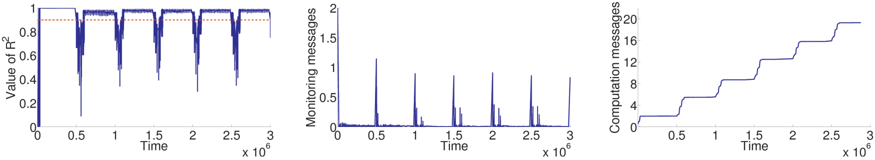


Figure: Accuracy and messages including convergecast

Application: smart grid CO_2 monitoring

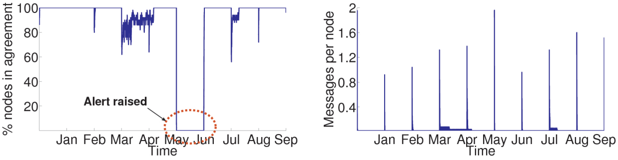


Figure: Accuracy and messages of *DReMo* for smart grid data monitoring

Conclusion

- First work on monitoring R^2 for distributed data
- Algorithm is provably correct, highly efficient and converges to the correct result very fast
- R^2 is scale-free
- Potential applications in many domains

References

- K. Bhaduri, K. Das, C. Giannella. Distributed Monitoring of the R^2 Statistic for Linear Regression. SDM. pp. 438-449. 2011.
- R. Wolff, K. Bhaduri, H. Kargupta. A Generic Local Algorithm for Mining Data Streams in Large Distributed Systems. IEEE TKDE. Volume 21, Issue 4, pp. 465-478. 2009.
- K. Bhaduri, H. Kargupta. A Scalable Local Algorithm for Distributed Multivariate Regression. SAM J. Volume 1, Issue 3, pp 177-194. 2008.

Algorithms for Speeding up Distance-Based Outlier Detection

Kanishka Bhaduri[1], Bryan Matthews[2], Chris Giannella[3]

[1]MCT Inc@NASA Ames, [2]SGT Inc@NASA Ames, [3]MITRE Corp

TU Dortmund Seminar

July 14, 2011

Roadmap

Given a dataset D:

t	$Speed$	$\dots$	Alt
1	100	$\dots$	1000
2	100.3	$\dots$	1002
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Distance-based outliers

Find all rows (instances) which are *outliers*

What is an outlier?

- Many definitions exist
- We use distance-based outlier

What is an outlier?

- Many definitions exist
- We use distance-based outlier

Distance-based outliers

A point is an outlier if it is very far from its nearest neighbors

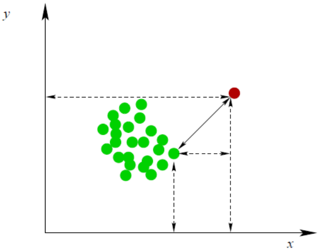


Figure: Distance-based outlier (red point)

Naive Approach

- For each point find the distance to its nearest neighbor (or k nearest neighbors)
- Sort the points in descending order based on this distance to its nearest neighbor
- These are the ranked outliers

Naive Approach

- For each point find the distance to its nearest neighbor (or k nearest neighbors)
- Sort the points in descending order based on this distance to its nearest neighbor
- These are the ranked outliers

Computational complexity

Quadratic with respect to the number of points

First improvement: Orca

- Problem relaxation: find only the top t outliers
- Algorithm maintains a cutoff threshold which is set to the score of the smallest outlier
- For each point, keep finding nearest neighbors until one is found less than the threshold
- Prune that point since it cannot be an outlier
- Disk-based implementation can handle any size of data
- Published by Bay and Schwabacher in KDD'03

First improvement: Orca

- Problem relaxation: find only the top t outliers
- Algorithm maintains a cutoff threshold which is set to the score of the smallest outlier
- For each point, keep finding nearest neighbors until one is found less than the threshold
- Prune that point since it cannot be an outlier
- Disk-based implementation can handle any size of data
- Published by Bay and Schwabacher in KDD'03

Computational complexity

On average, near-linear with respect to the number of points

Contributions

- Centralized iOrca improves upon the Orca using a novel indexing scheme
- Early termination criteria allows algorithm to stop without accessing all data points
- Distributed outlier detection algorithms much faster than existing methods

Speeding up Orca using Indexing (iOrca)

1. Update cutoff faster
2. Rearrange data in order to find the nearest neighbors approximately constant time
3. Avoid unnecessary disk access
4. Build index fast without loading all data in memory

1. Update cutoff faster
2. Rearrange data in order to find the nearest neighbors approximately constant time
3. Avoid unnecessary disk access
4. Build index fast without loading all data in memory

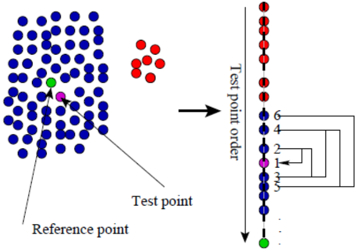


Figure: iOrca description

1. Update cutoff faster
2. Rearrange data in order to find the nearest neighbors approximately constant time
3. Avoid unnecessary disk access
4. Build index fast without loading all data in memory

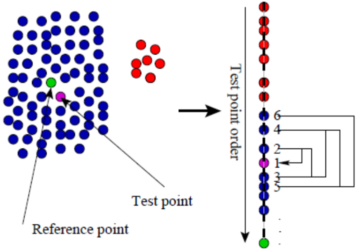


Figure: iOrca description

Can terminate even before looking at all the data!

Using distributed processing: iDOoR

- Nodes arranged in a ring
- Split data and assign to different nodes
- Central node ships blocks of test data for testing
- Test blocks proceed across the nodes in a ring
- Cutoff updated at end of each full round

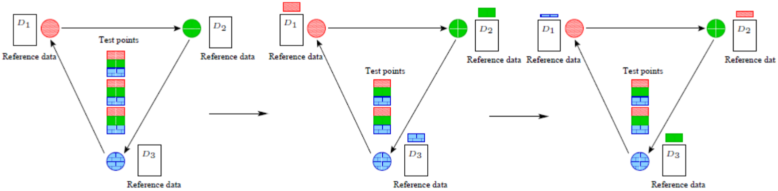


Figure: iDOor description

Experimental results: datasets

- *Covertype*: contains 581,012 instances and 10 features
- *Landsat*: contains 275,465 instances and 60 features
- *MODIS*: contains 15,900,968 instances and 7 features
- *CarrierX*: contains 97,814,864 instances and 19 features

Experimental results: performance of iOrca

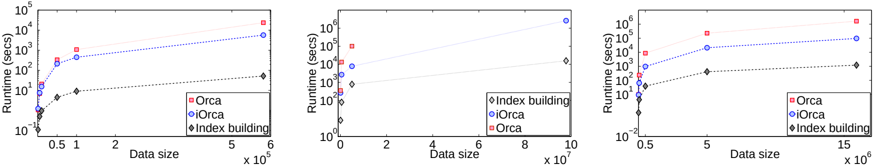


Figure: Running time of iOrca

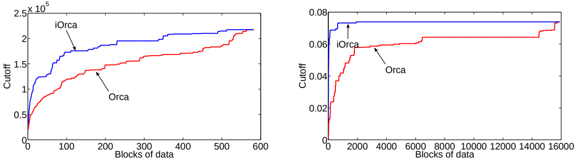


Figure: Cutoff increase of iOrca

Experimental results: running time of iDOoR

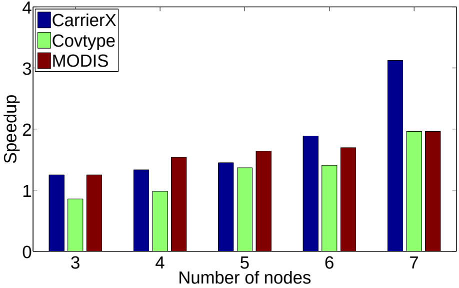


Figure: Running time of iDOoR vs iOrca

Conclusion

- Both sequential and distributed outlier detection methods that are significantly more efficient than existing methods
- Algorithms are provably correct
- *iOrca* and *iDOor* can terminate even before looking at all the test points
- Potential applications in many domains

Acknowledgements: NASA SSAT project

Resources:

- `http://ti.arc.nasa.gov/profile/kbhaduri/`
- `https://c3.ndc.nasa.gov/dashlink/`